THE SOFTWARE

HANDBOOK

For the VIC 20

David Thom *and* **Vic Numbers**

A PUBLICATION OF
PSIDAC

* SOFTWARE PIRATE'S HANDBOOK FOR C64 *

(BOOK TWO)
SPH-64

The SPH-20 (VIC-20) handbook has some information which applys directly to the C64. This info is covered in chapter one, three, and chapter four. The techniques will show you how to back up any cassette program written for the 20 or 64! Chapter four also gives schematics and layouts for making an interface for standard cassette recorders to the 20 or 64.

Duplication of cartridges and the use of disks is somewhat different on the C64, we are working on a "Book Two" (SPH-64). This new book will cover procedures for the C64 cartridges and disks. If you send a self addressed stamped envelope, we will alert you when the C64 book is available.

inf SPH-64 1283

SOFTWARE

PIRATE'S

HANDBOOK

for the VIC 20

DAVID THOM & VIC NUMBERS

ILLUSTRATIONS & COVER DESIGN BY
ACE CAMPBELL

PUBLISHER'S NOTE

This book is written as an information guide for those who wish to learn about and experiment with software protection and duplication methods. It is not intended to encourage theft or illegal uses of copyrighted software. The word Pirate is used only for it's literary value and should not be interpreted as meaning to steal software.

All information in this book is accurate insofar as can be determined by the authors and publisher. No liability can be assumed however, for any inaccuracies which may be inadvertently contained herein.

The user of this information must assume all liabilities associated with its use. The user must also assume all risk to person or property associated with the use of the circuitry described in this book. It is recommended that the user be technically competent to determine the suitability of the application. In no event shall the authors or publisher be liable for incidental or consequential damages in connection with the use of the information in this book.

VIC-20, VICMON, and VIC are trademarks of Commodore Business Machines, Inc.

Tapeworm, Romulator, and the "Software Pirate's" logo are trademarks of PSIDAC.

THE SOFTWARE PIRATE'S HANDBOOK

SPH-20 V883-3 Sugg. Retail $9.95

CONTENTS

INTRODUCTION

In modern society it is very difficult for the average person to keep up with technical changes and innovations outside their profession. As a result it is often profitable for those who do know the "secrets" of new technologies to attempt to restrict general access or knowledge. This is certainly the case in this "computer revolution" as you have probably already realized.

In the scope of this book we will try to provide some enlightment concerning the secrets as they pertain to the VIC-20. This information allows you to do programming using parts of the VIC-20 which for all intents and purposes would have otherwise been "off limits" or hidden. The VIC-20 certainly has a wealth of this "buried treasure." As with perhaps all knowledge there are good and fair uses of this information as well as illegal or unethical uses. As a user, you have the ultimate responsibility for the legality of application.

While on one hand we admit to a certain validity for maintaining "industrial secrets," we also feel that the state of the art can often be advanced by publication of ideas and information that becomes obvious to those of us who are interested enough to seek this information. If the secret of a skeleton key lock had been well kept, it might

still be the only lock available to protect your dwelling! TO NOT ATTEMPT TO UNDERSTAND TECHNICAL SECRETS IS LIKE WILLINGLY PUTTING ON A BLINDFOLD AND WALKING THE PLANK.

The PIRATES HANDBOOK begins by explaining some of the regulations, ethics, and accepted uses for the techniques of duplicating software. Chapter two deals with background on VIC-20 organization. VIC MON, the editor assembler, is explained with respect to it's ability to save machine programs. Chapter three explains concepts and theory of software protection techniques as well as ideas on the direction of future methods.

The last three chapters cover actual procedures for copying tapes and cartridges. These chapters can be used for quick referance once you understand the concepts outlined in the first three chapters. Appropriate software listings and schematics have been included where applicable.

We hope that this book will give you information and ideas that you can use to enhance the enjoyment of your computer. MAY THESE HERETOFORE UNCHARTED WATERS PROVIDE A VALUABLE VOYAGE.

CHAPTER ONE

CODE OF THE SOFTWARE SEAS

RULES AND REGULATIONS

You should never set sail without an understanding of the rules and regulations covering your activities. This chapter will explain the legal basis of software protection. Certain court cases have shaped the precedents which currently apply to software protection. We will also discuss the ethics and accepted practices for copying software.

Prior to 1980, software was virtually unprotected by U.S. law. You might say that it was a new kind of commodity in a new vessel. It was therefore unprotected simply as a result of the fact that it had not yet been given a legal definition in any of the laws that normally protect information. Software, being information, did not fall into the category of devices or processes which patent laws covered. Copyright law, which was the logical tool for protection, included works "formed by the collection and assembly of data" and "graphic works". These definitions did not specifically mention software or computer programs which might be stored and sold as the contents of an electronic device. The resulting confusion has persisted as the Pennsylvania hearing of the case of Apple vs Franklin Computer (E.D. Penn. 1982) demonstrated when the court concluded that there was "some doubt" as to whether unauthorized duplication of ROMs constituted copyright infringe-

ment.

The only effective legal means for programmers to protect their products and right as authors was to handle them as trade secrets. This meant that they needed elaborate methods to prevent the contents of their ROMs etc. from being read or copied. Furthermore, legal agreements had to be made between the seller and user that placed responsibility for unauthorized access of the software on the user. This actually limited the user in what, in any other product, would be their normal rights of ownership. Large software houses had another tactic, they did not sell software, they only sold a license to use it! In this manner the user could claim no rights of ownership.

Needless to say, this lack of legal definition probably was the primary carrier of the "Protection Paranoia" that is still an epidemic and is unparalleled in any other information format. I have yet to see a book with invisible printing!

In the Pre-1980 era, there were legal precedents which made protection a wise choice for software houses. In fact, the case of Tandy vs Personal Micro Computers Inc. (N.D. Cal. 1981), was the first time that a court held that ROM loaded with object code could be protected. The point to be taken here was that prior to this, ROM loaded software, was not only being copied, but also SOLD by other corporations!

it's really no wonder that Protection Paranoia was rampant!

The Tandy case was the first visible application of the medicine provided by the U.S. Congress in 1980. This was when they amended the existing copyright law to include "a set of statements or instructions to be used directly or indirectly in a computer." At last, software authors have a legal definition of their work and it is protected in the same manner as books, records, works of art, etc. All that is required to obtain this protection is to mark each software package with the © followed by the year of the first public distribution of their work and their name. Official registration has certain legal benefits but is not essential.

Really though, the software authors have much more protection than other types of writers because they can still use protection techniques on their works. This leaves the average person with no control over the way he uses the product. Software houses often claim that this protection is justified because they say any copying is illegal anyway. The fallacy here is that there are several reasons to make copies of software that have absolutely nothing to do with cheating the author out of a sale!

There are many times that a user may wish to have two or more programs in the computer at once and wish to switch from one to the other without turning off the computer, unplugging a cartridge and plugging in another. A simple, cheap way to

do this is with a switchable 16K (or larger) RAM with memory protect capability. (see chapter 3). No expander buses are required and much wear and tear is saved by simply switching 8K sections in and out of operation. With a disk drive and cartridges copied to your disk, you can have a virtual library of useful utility programs and switch from one to the other in seconds. Power requirements would not simply allow this to be viable in an expansion bus scheme.

If the user of a word processor program wishes to use a printer, other than that the program was written for, he must be able to modify certain routines to suit his needs. With protected software, he really has no choice in the matter and may find that his computer use has been needlessly restricted. With locked up, protected, undocumented software, the average users are forced into equipment/useability limitations simply because of the difficult task of "unlocking" the software to make changes. How would you like it if a car manufacturer put special locks on the wheels so that only certain garages could change the tires, charging whatever THEY decided was a fair price! You probably would never buy such a car and if you did you would feel compelled to defeat the locking system!

Many products require documentation to be fully useful. There are many of these types of products which at one time were attempted to be protected by witholding access of important product documentation. In these cases the useabil-

ity and thereby, the ultimate marketability of the product was greatly limited. There is in fact, the legal aspect of "merchantability" of a product which in general means that the purchaser of a product has the legal right to expect that product to be able to perform the normal functions that are associated with that type of product. This is rather nebulous as it might apply to software but it is designed to protect the purchaser from shoddy products or those that are unreasonably limited. It seems as though if you buy a computer or software you should not have parts of it "locked out" just to keep you from full use of it.

Time and again various industries have gone through this "secrets" game only to find in the end that a large number of the consumers have a right, need, and demand to know what's inside. Limiting this information can only limit the useability of the equipment and also limits the growth of add-ons which make the original product more useful. The VIC-20 is better than most computers in this respect. They have made schematics and detailed information about their operating system available to everyone. This makes the VIC-20 a very useable computer. It's doubtful that anyone is building VIC-20s in their garage for "black marketing". They couldn't compete! Software houses should perhaps take a lesson here. Well written, well documented software sold for the price of a record album or book, would be so popular that everyone who needed it would want

it. Furthermore the price and hassle of photocopying the documentation would make it ridiculous to Pirate it!

In a federal court case, Williams vs Artis International (Federal Appellate Court 3rd Cir. 1982) Artis was enjoined permanently in the sale of unauthorized "Defender" video game ROMs. As in the Tandy case this strengthens the legal precedent concerning the illegality of "copying for sale" of software. Software authors have this protection and should use it.

If you intend to Pirate for sale or distribution to others, you will be performing a criminal act. That is, stealing the work of others and selling or distributing stolen goods. If however, you have a private need such as modification of software or more versatile access from disk or RAM or making backup copies, you will have a need to "Pirate" programs. Endeavoring to gain full use of software you own does not seem to be a criminal act!

CHAPTER TWO

MAP-O-THE-20

MEMORY MAPS

The next thing that you will need for your voyage on the software seas, is a good set of charts. Software is loaded into and operated from memory, and memory is located and utilized via addressing. This is not unlike places on the globe which are located by addresses. This chapter will provide you with maps and the necessary chart skills to assist your navigation.

When you first turn on your computer and it says "READY", you would generally say that memory is empty and you can then fill or LOAD memory with your program. In the purest sense though, this is incorrect. When the computer is first turned on, one of the very first things it does is to generate addresses of locations in memory, and then to execute the instructions there. These instructions are located in ROM inside your computer and are called an "operating system" (VIC-20 uses the nickname "KERNAL"). In essence, the computer is walking itself through this operating system program to establish certain characteristics of its operation. This program does many things other than generating the ready message and header. Most of the Kernal features are very handy most of the time. At some times however, it becomes the primary source of some of the limitations of the VIC-20. One of our goals is to "fool" the computer in such a way

that the Kernal does not take control and do things that we don't wish it to. You might think of this as a way of gaining manual override control when desired. For example, when you plug in a cartridge and turn on the computer, usually the cartridge program begins to run. You are then unable to list, save, etcetera. The manual override system can prevent this from happening so that the user still has some control over the system. Without this manual override, you are being restricted in your full access to your computer!

MAPS

The maps in this section will provide you with a means of visualizing the sections of the computer so that you can better understand how it all works.

First of all, you should realize that there are three number systems that are commonly used to identify locations and information in the computer memory. Of least concern to us is binary (base two). Let it suffice to say that in this system all numeric values are represented by ones and zeros and it is the "hardware language" of the computer. That is, computer chips are generating and responding to combinations of HI and LO voltages (ones and zeros) in the process of running a program.

You would probably find binary to be very unwieldy. You are used to working with decimal (base ten). Any number which can be represented by the computer hardware in binary

FIGURE 2.1

VIC-20 MEMORY MAP

DECIMAL			HEX
0	WORKING RAM AREA 1K USED BY VIC	1K	0000
1024	3K EXPANSION AREA *	3K	0400
4096	3583 BYTES OF USER RAM (IN VIC)	3K	1000
7680	SCREEN RAM (ON UNEXPANDED VICs)	512	1E00
8192	8K EXPANDER BLOCK 1 *	8K	2000
16384	8K EXPANDER BLOCK 2 *	8K	4000
24576	8K EXPANDER BLOCK 3 *	8K	6000
32768	OTHER VIC FUNCTIONS: CHARACTER GENERATOR ROM, COLOR MEMORY, I/O etc.	8K	8000
40960	8K ROM CARTRIDGE AREA BLOCK 5 *	8K	A000
49152	BASIC AND KERNAL ROM	16K	C000
65535			FFFF

* Not contained in unexpanded VIC.

form also has a decimal equivalent which is much easier for us to manipulate.

Finally, there is the hexadecimal (base sixteen) number system. It is widely accepted for computer use because the conversion process from binary to hexadecimal and vice versa, is a very direct and simple one. Appendix A gives a more in depth study of number systems and conversions. Also a program is listed there called HEX-DE-CON which can be used for calculating equivalent values.

In this book we will normally use decimal. When the situation requires a hex value for simplicity, we will identify the hex number with a preceeding dollar sign (i.e. $1C00).

Study the map in figure 2.1. You will note that a large portion of the 65,536 bytes of memory are used by the VIC (shaded area). These areas are available to the user only to the extent that you can access some of the routines stored here, and that certain locations can be peeked or poked to obtain certain information or results.

An unexpanded VIC contains all except the starred memory locations. Thus the VIC RAM (3583 bytes) is the only spot available in an unexpanded system for user programs or taped software. The available RAM area can be expanded in 8K "blocks". Each block actually contains 8,192 bytes or

locations capable of storing eight bit characters. (see appendix A for bits and bytes). The VIC can have three of these blocks added (24K) and for certain uses an additional block (#5). The 3K area has very limited use due to the manner in which the screen is relocated when expanding memory.

There is a hierarchy for the use of the memory which we will occasionally change to implement certain copy procedures. For basic programming, RAM must be added in sequence. Block one, block two, and finally, block three. The 3K expander resides in front of the VIC RAM. Block five is designed as a ROM ONLY memory location. The Kernal won't allow you to write basic programs in block five or to save a program located in block five. Be sure that you understand the concepts of blocks and the map before you continue. A good deal of the pirate techniques involve a process called "block switching".

CARTRIDGE LOCATION

Block five is intended for cartridge software (also called firmware). Chapter six deals extensively with the technique of using RAM loaded with copies of ROM programs in block five. We will fool the computer into thinking it has a normal commercial cartridge plugged in! In order to make successful copies of these cartridges, we must be able to determine their usual location in memory.

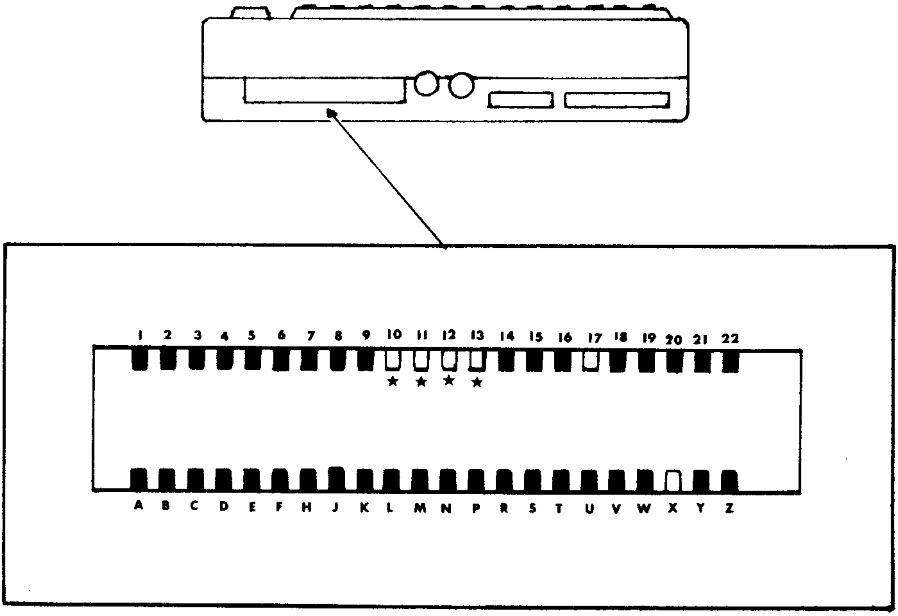
When power is applied to the VIC, the Kernal looks for a program in block five. If a cartridge is there, the correct information is found and the program will begin operation. By design, you will have no user options other than those provided by the program. This is however a simple method of determining if a cartridge contains a block five program. If it runs on power up, at least block five is being used. We say "at least" because it could contain more than just block five ROM. Many programs require more than the 8K available in block five. It seems that the general trend when programs are larger than 8K is to use blocks five and three. There is really no hardware restriction on which blocks a cartridge is designed to use. The five and three convention just seems popular. YOU, however, are restricted and must use the software in the blocks it was designed for. This is true regardless whether you are using the original cartridge or a copy loaded into RAM.

All of this means that if you wish to copy a cartridge, you MUST determine where its ROM is located in memory. The program "Block Check" and instructions for use are located in chapter six. This program can be used as an aid in finding out ROM locations in a cartridge.

Any cartridge that uses a "SYS" number for starting can be located by comparing that number to the memory map and seeing which block that number would be located in. The "SYS" number does not have to be at the beginning of a block.

FIGURE 2.2

EXPANSION PORT EDGE CONNECTOR



PIN#	FUNCTION
10	Block 1 Select line
11	Block 2 Select line
12	Block 3 Select line
13	Block 5 Select line
17	Write line (VR/W)
X	RESET line

White pins indicate computer lines which are used for block switching. This process which is described in detail in chapter six, is the basis for making copies of cartridge software. Also shown are the RESET and Write lines which provide coldstart and ROM emulation capability respectively.

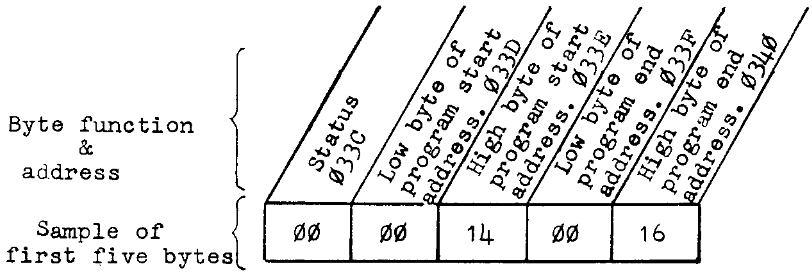
Machine language programs can be written to begin at virtually any available memory location. Occasionally the "SYS" number will be written as a formula instead of a single value. You must simply calculate the value and look it up in the table. For example, a "SYS" 7*4096 starts at location 28672. You will note from the memory map that this location lies in block three. Assuming that the ROM crosses no block boundaries, we would consider this program to be a "block three program". Remember that this is true regardless of how much of block three it actually occupies. We consider only whole blocks.

In some tough cases you might need to open a cartridge up and examine the wiring to the edge connector to determine which blocks are being used. Use figure 2.2. Be sure that the circuit card is correctly oriented. Notice the starred (*) pads. These are the memory block select lines. (or RAM select for 3K expansion area). By determining which of these are connected to the chips, you can ascertain which blocks are being used.

In summary, cartridges contain ROM, usually in block five but potentially in any free block. The memory can be located by: (1) checking for auto start (block five), (2) using "Block Check" software, (3) determining location by "SYS" number, or (4) checking the physical wiring. Once you have determined where a cartridge resides, the Romulator System (chapter six) can be used to save it to tape or disk.

FIGURE 2.3

TAPE BUFFER



Identification of the first five locations of the tape buffer. (Ø33C - Ø34Ø) All values in Hex.

TAPE BUFFER MAP

HEX	
Ø33C	STARTING & ENDING ADDRESS & STATUS BYTE
Ø34Ø	16 BYTES USED FOR PROGRAM NAME
Ø35Ø	REMAINDER OF BUFFER USED TO STORE TAPE DATA DURING TRANSFER FROM TAPE TO VIC
Ø4ØØ	

SAMPLE OF VIC MON BUFFER DISPLAY

```
B*
    PC  SR AC XR YR SP
.;603E 33 00 63 00 F6
.E1DØØ
.MØ33C Ø35Ø
.:033C 03 00 14 00 16
.:0341 47 4F 52 46 00
.:0346 00 00 00 00 00
.:034B 00 00 00 00 00
.:0350 00 20 20 20 20
```

TAPE SOFTWARE

Programs that you purchase on tape have an interesting way of using memory. The recorded leader of the tape contains information telling the computer where to load the program. For basic programs, this is pretty well fixed. Machine language programs however can be written to run starting virtually anywhere in the unshaded areas shown on the memory map.

The VIC has a 192 byte memory slot or "buffer" which is used for cassette data. This buffer is located starting at location 828. The second through the fifth byte contain the starting and ending address of the program being loaded from tape. This information comes directly from the recorded leader of the program tape. Figure 2.3 shows the exact organization of the tape buffer. If you wished to copy a machine language program to tape you must be able to locate it by looking at the tape buffer. However, SAVMACH which is listed in chapter five can be used to save a machine program without knowing the starting and ending locations.

VIC-MON by Commodore is an invaluable addition to your arsenal. It is a machine language editor-assembler for the VIC-20. It can often help in locating programs and performing other operations outside the jurisdiction of the Kernal. If you are a beginner to programming, you may wish to stick to the canned routines which we have provided. If however you wish to do machine language programming and really want

to get control of the nooks and crannies of your VIC, an editor assembler is the thing for you. Chapter six techniques can be used to make a copy of VIC-MON on tape or disk.

There are many complete books on machine language programming so we will not try to rehash that here. The features of the editor assembler that we are most interested in are its abilities to circumvent the limitations imposed by the Kernal for copying software. To be successful in making copies of your software, you don't need to know machine language programming. A few of the operational features of VIC-MON (or equivalent) will help you greatly. Of primary concern to us is the ability to display or modify the contents of the tape buffer, and to save machine language programs.

For those who are unfamiliar with VIC-MON we will list this process here. From then on this book will simply mention for you to examine the tape buffer or perform a machine language save.

EXAMINE TAPE BUFFER

1. With VIC-MON located in block three, type SYS24576 RETURN
2. The VIC-MON header

   B* PC SR AC XR YR SP   should appear
3. Type E1D00 RETURN
4. Type M033C 0350 RETURN

5. The display now show the HEX values of the numbers in these locations. Figure 2.3 explains what they are.
6. For example, the values shown in figure 2.3 starting at 033C mean that the program begins at location $1400 and ends at $1600.
7. The next few lines on the screen contain the hex equivalents of the ASCII values of the letters in the program name.
8. Type X and hit RETURN to escape VIC-MON.

SAVE MACHINE PROGRAM

1. Repeat 1-3 above
2. Type S, "Prog name", device #, start address, end address then hit RETURN.
3. Example: S,"TEST",01,1400,1600 RETURN
4. Saving message should appear.
5. Remember machine language programs require this type of save or the use of SAVMACH (chapter five).
6. Machine programs can be usually identified by their requirement for a LOAD"",1,1 command, or the inability to list them.

CHAPTER THREE

HIDDEN TREASURES

SOFTWARE PROTECTION CONCEPTS

In this chapter we will discuss a few of the most commonly used methods for protecting VIC-20 software. We will also more generally discuss concepts that will give you a better idea of what may lie ahead in the future of software protection.

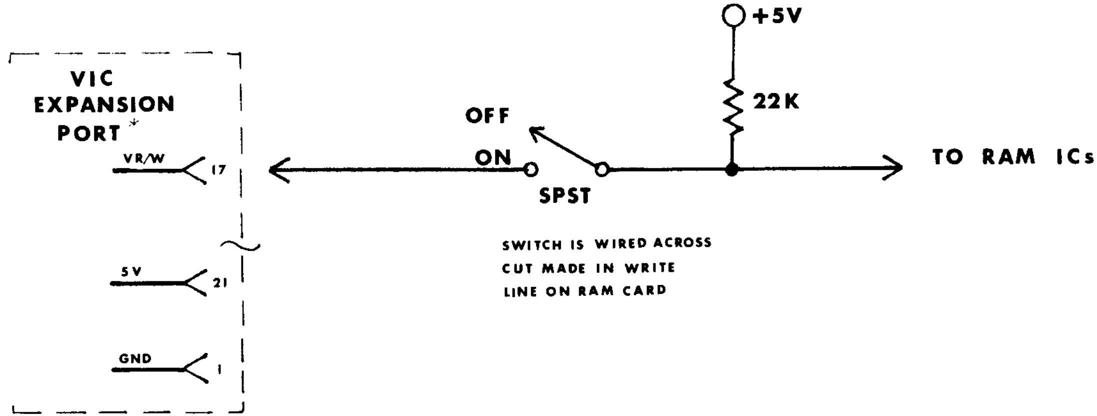
One of the key concepts of protecting software follows a proven and age old method for protecting any valuable, that is, put it where it is hard to find. In short, HIDE IT! This is a form of what is being done in cartridge software. The Kernal does not allow you to save from the locations where the cartridge normally resides. The program is effectively "hidden". Machine language tapes also use this technique. When the tape is loaded, the recorded leader gives the starting address of the program. The program is then loaded into the locations following that address. If you don't know how to use the tape buffer, you can't find the program to save it!

CARTRIDGE PROTECTION

Chapter two gives quite a bit of information concerning how to find the location of a cartridge. Once it has been found, chapter six procedures tell how it can be saved to tape or disk. To fully appreciate why some of the existing

FIGURE 3.1

WRITE ENABLE SWITCH



This modification on memory cards will allow the RAM memory to be switched from normal to "ROM" emulate. With the write enable switch OFF (as shown), the RAM cannot be written to and thus looks like ROM to the computer. It is usually a simple matter to locate the write line on your RAM card leading from edge pad #17. A cut is made in this trace before it leads to the ICs. The switch and 22K ohm resistor are then wired as shown. Chapter six has complete information on use and a PC card layout for modifying Commodore RAMs.

* See figure 2.2.

protection techniques have been selected, it is helpful to understand how RAM can be used to run taped copies of programs.

Tape or disk copies of programs can be down loaded at will into RAM which is located in one of the three unprotected memory blocks. Then by switching the RAM into the location the cartridge was designed for, we can restart the computer and run the program. (using coldstart not power up) The coldstart forces the Kernal to begin over as though if the computer had just been turned on but without the associated power interuption and RAM data loss.

One trick used to foil this method is for the program to contain a routine that will try to write garbage over the program when it runs. Assuming it is in ROM, this would have no effect, the computer cannot rewrite ROM. If however this was a pirated copy in RAM, the program would destroy itself. This is like "haunted gold", you can dig it up but don't try to use it!

There is a way to get rid of this curse. That is to install a switchable WRITE line on your RAM. Figure 3.1 shows the schematic for wiring. The function of the write switch is to physically cut off the computer's ability to write to RAM whenever we want the RAM to look like ROM. Turning the write line off causes the RAM to effectively emulate ROM. Thus we have entitled our ROM emulating system

ROMULATOR. Of the many cartridge games and utilities that we have encountered, all of them could be duplicated with Romulator techniques. (chapter six)

It is possible and likely that programmers will turn to other more diabolical protection schemes. If they feel threatened by this kind of information. It is probably easier to devise protection schemes than to break them. Consider that to the designer there are unlimited possibilities to choose from, to the Pirate, this means unlimited obstacles to overcome. It will be helpfull then to also consider a couple of ideas for protection which may surface in some form in the future.

One such technique which would be difficult to circumvent would involve the use of a small RAM located arbitrarily in another block. The RAM would only need to be a few bytes in size. Upon running, the program would write a secret access code to several locations including the one which contains the hidden RAM. The routine would then check each location it had written to. It would cause a default if it found the access code in any wrong location or if the access code wasn't in the correct location. This would force the Pirate to determine the exact size and location of the key RAM and to hardwire a circuit in the proper location. An attempt to use a full block of RAM would allow the access code to show up in the wrong locations causing a default.

The second future protection concept we will describe works on the same principle as many industrial program fail-safe systems. That is, that as long as the program is run - ing, a circuit is periodically signaled. If this signal is present, the computer can run uninhibited. If the program stops, goes beserk,or attempts mutiny, the failsafe will lose its periodic signal and force a shutdown.

The VIC provides enough versatility at the expansion port that this could be done in several ways. The simplest and one that I might choose would involve a fifty cent add-ition to the cartridge. This system would use a monostable timer which would require a periodic pulse on one of the I/O select lines (generated by a running program). Loss of this would, after a few seconds, cause the monostable output to force a coldstart, or lock up an address line, or some similar shutdown scheme. Since the program cannot run while it is being copied, the failsafe circuit would lock-up the computer during the copy attempt. Any attempt to copy would require a detailed understanding of the circuit interaction.

These are only two ideas out of hundreds of possibilit-ies. There are bound to be some very effective protection schemes in the minds of programmers. But by waiting for the software market to mature, there are many cartridges which are already on the market or on their way. Last minute changes are costly, especially if they involve hardware. You can be assured that the copy procedures in the next three

chapters will work on a substantial variety of "present technology" cartridges.

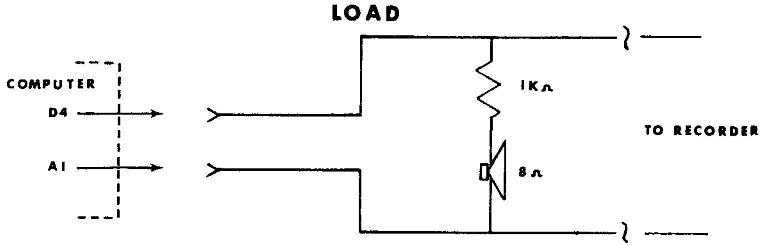
It is probable that future software will involve some sort of access codes and hardware protection systems. Here again the problem will lie in the fact that if the user knows the code locations and can understand the hardware systems, he will have little difficulty in duplicating the software. It really seems silly to attempt protection on the basis that the general public is too "dumb" to learn how to defeat that protection. It is also probably criminal to try to suppress the free exchange of the information and knowledge of such systems to the general public. I think we have a case of Prima donnas, who feel that because they were born into an era when we write with computer keyboards and store the results of our creativity in microchips instead of on paper, that it is all more sacred and valuable than any ordinary printed work. We should all hope for a time when software becomes such a prolific commodity that prices and availability are in line with the rest of the information market.

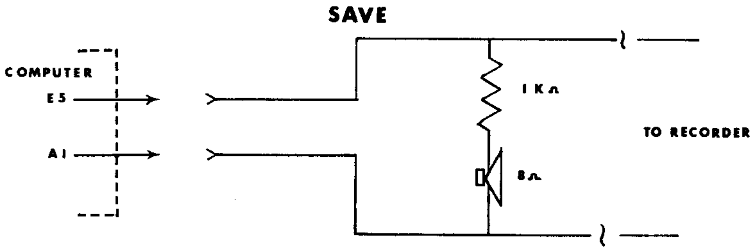
TAPE SOFTWARE

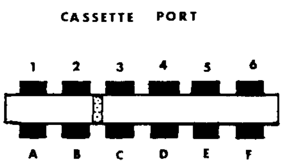
Tapes typically use several different protection methods. They are not quite as predictable as cartridges. Most seem to revolve around "trashing" the tape buffer, disabling the keyboard functions, and self-destructing if certain LOAD RUN sequences are not correct.

FIGURE 3.2

LOAD DATA AUDIO & SAVE DATA AUDIO







PIN #	FUNCTION
A-1	Ground
B-2	+5 Volts
C-3	Cassette Motor
D-4	Cassette READ
E-5	Cassette WRITE
F-6	Cassette Switch

The LOAD data audio circuit provides an audio output during LOAD operations. This is useful in determining characteristics of pre-recorded program tapes. It also provides a simple way to align the tape head by "ear". (Chapter four - Head alignment procedure). Installation can be in the computer or on lines D-4 and A-1 where they enter the datasette. If you are using a Tapeworm, or similar interface, parts can be mounted on the interface unit itself. (Use earphone for speaker)

The SAVE data audio circuit is primarily for "Header Swapping" which is detailed in chapter five. You may choose to wire two aligator clips to an earphone with a 1K ohm series resistor. In this manner, you can simply clip it across E-5 and A-1 when you are performing header swapping. The SAVE data audio circuit provides audio only during the time that the computer is saving data to tape.

In the case of machine tapes you must use the machine save routine as described in chapter two, or SAVMACH (chapter five). The procedure is to first load the program in question. Do not run it. Next examine the tape buffer and then perform a machine save. Be sure to use the addresses you read from the tape buffer for saving. Another possibility is that the tape buffer will be destroyed during the run process. In this case you should stop the tape immediately after the recorded leader has loaded. (This is when the FOUND message appears). You can then examine the tape buffer to see where the program is going to be put. This way, no matter what they do to the buffer later on, you already know where the program is going to load to. It is very helpful to have an audio output on your datasette so that you can hear the data and distinguish exactly when the data stream begins or ends. Figure 3.2 details such a circuit.

Taped software can be protected fairly well by using a multiple load scheme. In this method, several chained programs successivly load the next. By requiring certain poke statements to be executed, each segment is required in order to run the next. If a save is attempted after loading the program, only a part of the program would be saved and therefore be useless. This is a case where the audible data circuit will be invaluable. With a little practice you will be able to hear the difference between the tone leader, header, program, and data files.

On this type of program you can stop it after each load and then try to break down each program unit to see if it is copiable. Remember that each program section will have a tone leader followed by a very short data burst, (header) and then another tone leader followed by a longer data burst, (program). This is when you need to stop the recorder to examine the program.

If the program does not make any sense, it may be a machine language routine inserted between basic programs. Be alert for LOAD"",1,1 commands in any of the basic routines. This is a dead giveaway that a machine routine or machine formatted data is to follow. Remember that you must use a machine save on machine routines. (chapter two and five)

One interesting protection method involves loading data into the screen memory. When these programs run they look at the screen memory for a particular pattern before they will operate. This pattern may be invisible since the color can be the same as the background. The old trick of black on black for invisibility! If you run into a program that causes information on the screen to change or disappear, this protection method is probably being used. Avoid using the "SHIFT/RUN" feature of the VIC when trying to break down these programs (and most others). You can usually find the secret contents of the screen by changing the background colors and moving the cursor around the screen. The data then becomes visible. In a few cases you may need to examine

the contents of the screen memory with the VIC-MON memory dump or an equivalent method. By checking memory before and after loads, you can spot any changes. You must be aware that if you copy these programs, they won't run unless the screen contains the exact information their load produced. Usually this means CLEARING the screen and typing in the correct information.

DUMB COPIERS

If someone gave you a sheet of paper with an encoded message, you would not need to be able to understand or even identify the enciphered characters to copy the message. We call this DUMB copying or CLONING. A grocery store photocopy machine could make a near perfect replica of the enciphered message while the worlds greatest deciphering system might not be able to break the code. Likewise, you really don't need to be able to break protection codes if you want to copy a tape. If you have a dumb copier you can transfer the program to another tape very easily. Although there is usually some degradation between copies, you should have no problem if you always work from an original.

Dumb copiers are a little more than an audio to audio copy. Usually plugging two cassette recorders together to duplicate tapes will not work. This is due to the limitations of audio recorders. Chapter four describes a dumb copier system which restores the digital level to the data, and conditions it, for the duplicating recorder. This system will provide you with quality clones.

CHAPTER FOUR

SOUNDING THE DEPTHS

AUDIO DUPLICATION OF TAPES

In these last three chapters we will be concentrating primarily on procedures and tools required to duplicate commercial programs. You can probably use these chapters effectively without an understanding of what is being done. However if you wish to know more about the theory behind these procedures, you should review the appropriate sections in chapters two and three.

Chapter four features TAPEWORM (tm) which is a cassette interface for standard recorders. Audio duplication requires two cassette decks and even if you already have a datasette, you may not wish to purchase another for duplication purposes. The tapeworm allows you to use most standard recorders in place of the datasette. The tapeworm can be used with a standard recorder instead of a datasette and it provides the basis for a dumb copier.

CLONEPLUG (tm) is a simple adapter plug that works with tapeworm to facilitate the dumb copier. (The Tapeworm isn't required if you have two datasettes. See figure 4.7) The Cloneplug system can be used to make copies of virtually any program tapes. The copies will load and run identically to the original.

TAPEWORM

DESCRIPTION

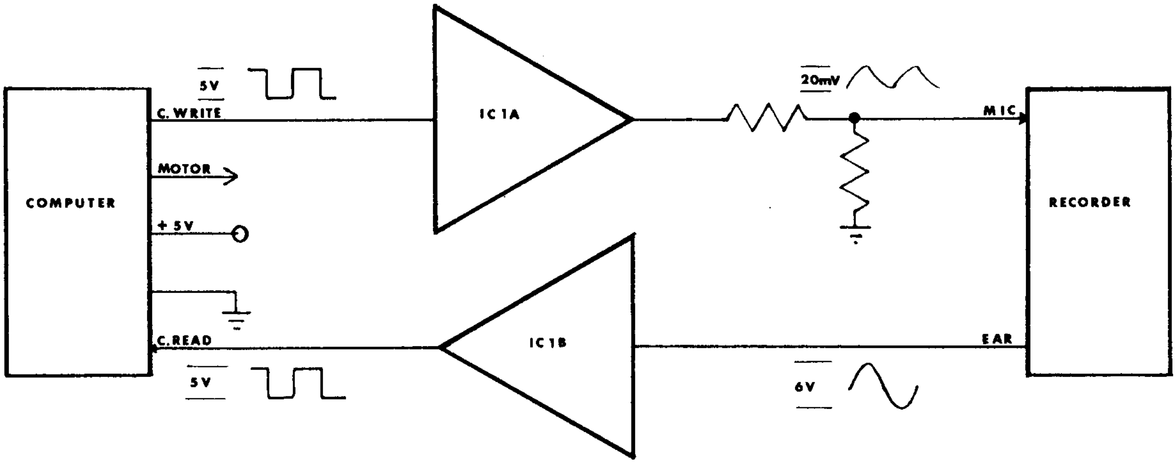
The TAPEWORM provides the proper interface circuitry between the VIC-20 and -64 series computers and most standard tape cassette recorders.

FEATURES

- The TAPEWORM is an inexpensive reliable alternative to the purchase of higher priced single use cassette data recorders.
- When not being used with the computer, your cassette recorder can be used for normal recording applications.
- TAPEWORM allows the computer to control the cassette recorder to play and record voice/sound information under program command; i.e., telephone answering-security monitoring-slide show sound-etc.
- TAPEWORM allows manual adjustment of the volume output level of the cassette so that you have the ability to compensate for tape quality variations.
- With an optional modification, the data can be heard during load operations.
- Recommended tape recorder: SANYO SLIM 1 or SLIM 2. Other tape recorders may work with TAPEWORM. Variations in record levels, fidelity, 6VDC adapter plug polarity, voltage, etc. between manufacturers requires some technical discretion before making cassette recorder substitution.
- A.C. adapters are not needed. The cassette and TAPEWORM obtain 6VDC power from the computer.

FIGURE 4.1

TAPEWORM BLOCK DIAGRAM



Computer output and input must be 5V square waves. When recording, the Tapeworm conditions the computer signal to feed the MIC input on the recorder. When playing, IC1B circuitry restores the digital level to the EAR signal from the cassette.

- TAPEWORM used with CLONEPLUG facilitates tape duplication using standard recorders.
- Simple to hookup, ear, mic, and power plugs provide all cassette interface connections.

TAPEWORM THEORY OF OPERATION

Refer to figure 4.1 for the block diagram of Tapeworm function. Both the cassette write and cassette read signals for the VIC are five volt square waves. Cassette recorders do not handle square waves well, they are much better suited for sine wave inputs and outputs. Furthermore, a microphone input on a cassette recorder expects to see a 10 to 20MV signal, not 5V.

The input circuitry to the cassette is made up of IC1A and the voltage divider consisting of the 1K and 100 ohm resistors. IC1A is designed as an integrator which rounds off the 5V square waves from the VIC. The voltage divider then provides the proper level of about 20 MV for the MIC input of the cassette deck. The sketches of the signals on figure 4.1 show the approximate conditioning taking place.

Since the output of the tape recorder during play is a sinewave, the output circuitry consisting of IC1B and Q1 must provide a 5V square wave to the VIC. IC1B is designed as a high gain clipping amplifier, Q1 provides a fast risetime 5V squarewave which is fed into the cassette read line of the VIC.

The cassette switch line is grounded to eliminate the need for wiring inside your cassette recorder. The computer supplies all power to the IC and cassette motor.

ASSEMBLY

Figures 4.2-4.4 give the schematic and layouts for Tapeworm. Appendix B lists materials available from PSIDAC or you may use your own resources.

Install parts as shown in figure 4.4. Notice that the six pin edge connector is soldered directly to the PC board.

The MIC ground should be left open on one end to prevent ground loop interference.

An optional LED modification shown allows you to "see" the data as it loads.

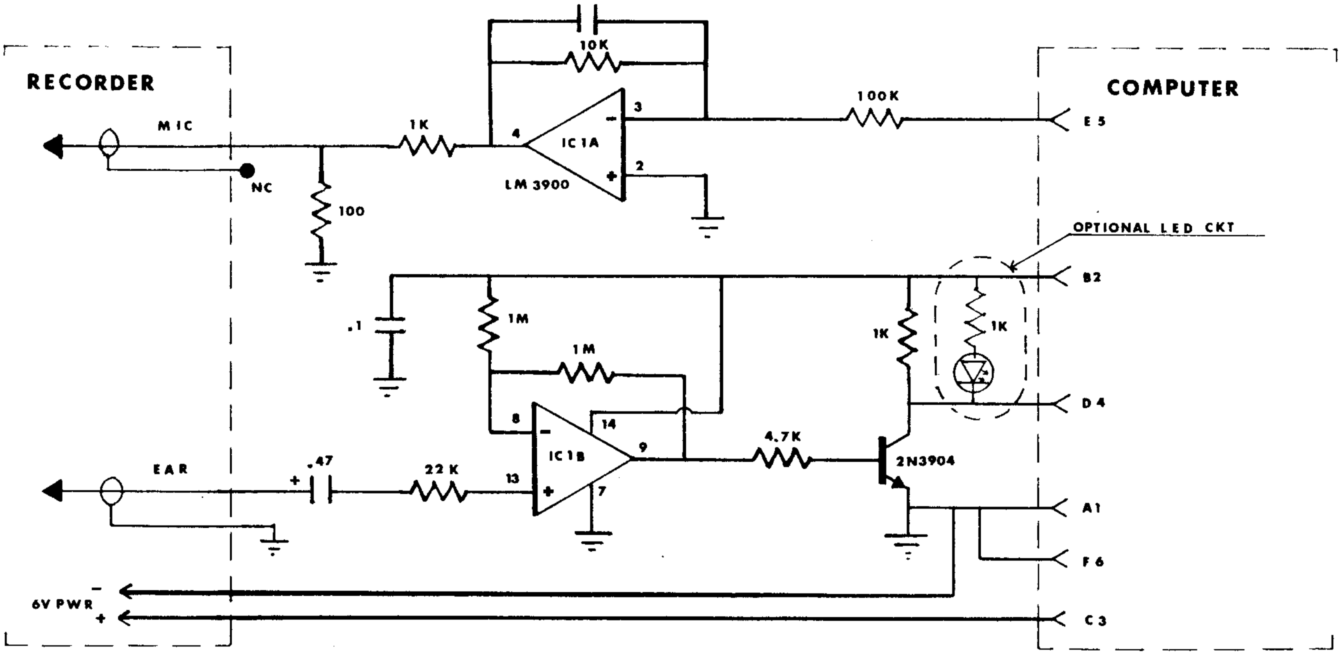
POLARITY

The power plug should be wired in accordance with your recorder. Most have the negative on the center pin. Sanyo is opposite! See figure 4.4 inset.

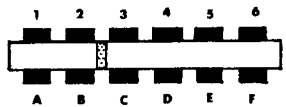
** Tapeworm cannot be used with positive ground recorders! Be sure to verify this. ** (some Panasonics)

If you make a mistake on power connections it will blow the computer fuse. It is highly unlikely to cause any other damage.

FIGURE 4.2 TAPEWORM SCHEMATIC

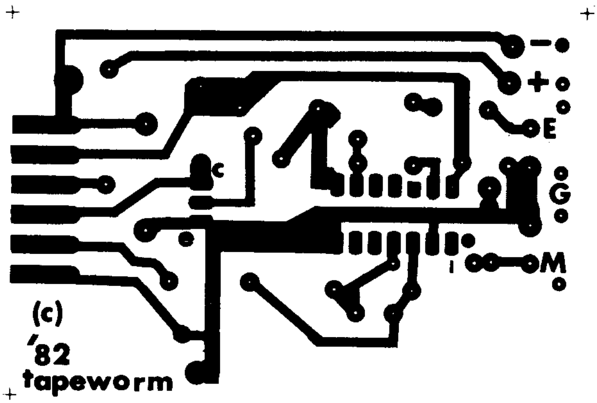


CASSETTE PORT



PIN #	FUNCTION
A-1	Ground
B-2	+5 Volts
C-3	Cassette Motor
D-4	Cassette READ
E-5	Cassette WRITE
F-6	Cassette Switch

FIGURE 4.3
TAPEWORM PC LAYOUT

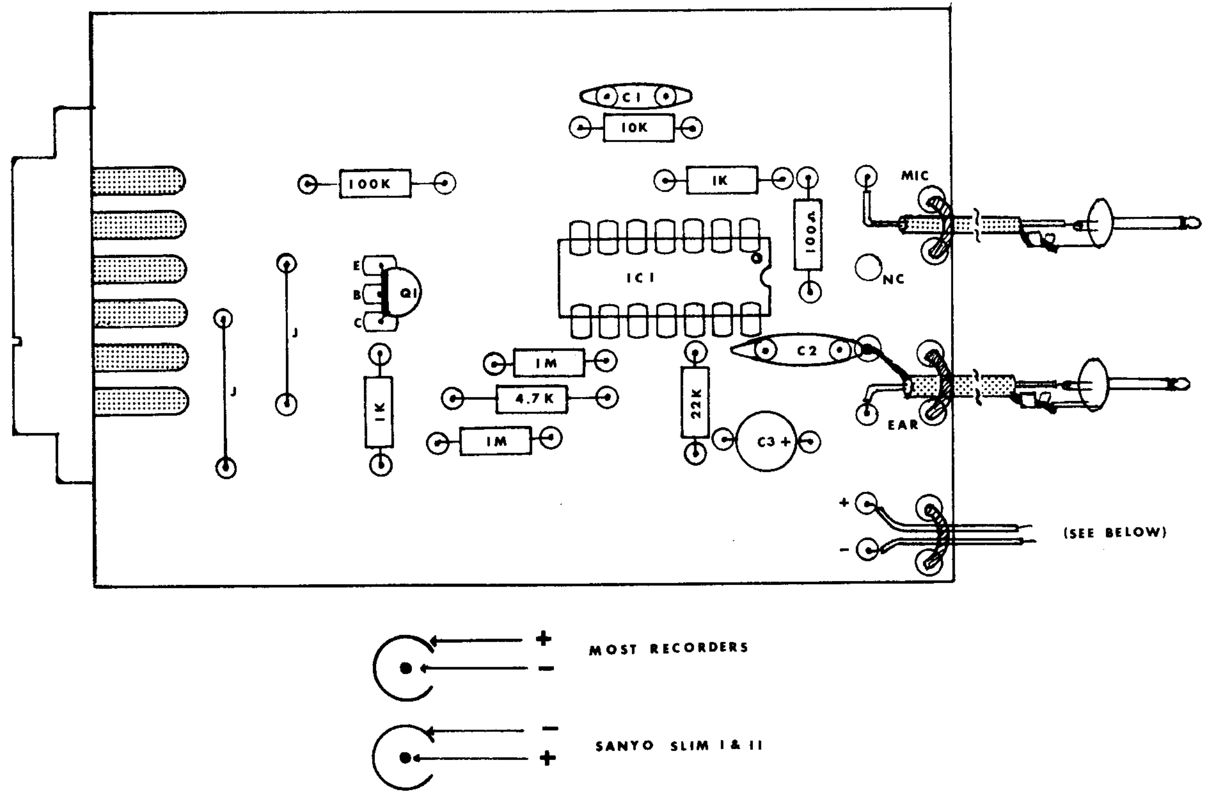


1:1 POSITIVE PC LAYOUT (FOIL SIDE SHOWN)

PT#	QTY	DESCRIPTION	RADIO SHACK EQUIV.
C1	1	.0047uf Disc Cap. 12V	272-130
C2	1	.1uf Disc Cap. 12V	272-135
C3	1	.47uf Electrolytic 16V	272-1417
IC1	1	LM3900 Quad OP Amp	276-1713
Q1	1	2N3904 NPN Switching	276-2016
R1	1	100 ohm ¼ W resistor	271-1311
R2-3	2	1K ohm ¼W resistor	271-1321
R4	1	4.7K ohm ¼W resistor	271-1330
R5	1	10K ohm ¼W resistor	271-1335
R6	1	22K ohm ¼W resistor	271-1339
R7	1	100K ohm ¼W resistor	271-1347
R8-9	2	1M ohm ¼W resistor	271-1356
EC1	1	6 pin .156" edge connector	(PSIDAC appendix B)
P1-2	2	1/8" Mini phone plug (3.5mm)	274-286 (or -287)
P3	1	DC power plug (to match your recorder)	274-1551
Misc.		Wire ties, mini coax, solder, etc.	

★ For complete kit, PC board, or parts, see appendix B ★

FIGURE 4.4
COMPONENT LAYOUT



★ DETERMINE CORRECT POLARITY ON YOUR RECORDER

HOOK UP

- Always plug the TAPEWORM into the computer TOP SIDE UP with the computer TURNED OFF!
- Make sure all cassette recorder switches are up or OFF before switching the computer on.
- Use high output low noise tapes of good quality.
- Insert 'MIC' and 'EAR' plugs into cassette jacks marked 'MIC', 'EAR'.
- Insert TAPEWORM plug marked DC6V into cassette jack marked DC 6V.

OPERATION

- Turn on computer.
- Type SAVE (or S shift A) then press Return.
- Press Run/Stop.
- The cassette recorder volume should normally be set to about 3/4 of full volume. This setting may vary depending on tape quality and the recorder used.
- The computer SAVE, LOAD and VERIFY operations will now function in accordance with the computer instruction guide.

  NOTE: The computer "PRESS PLAY ON TAPE" and "PRESS PLAY AND RECORD ON TAPE" messages will not be displayed when using TAPEWORM.

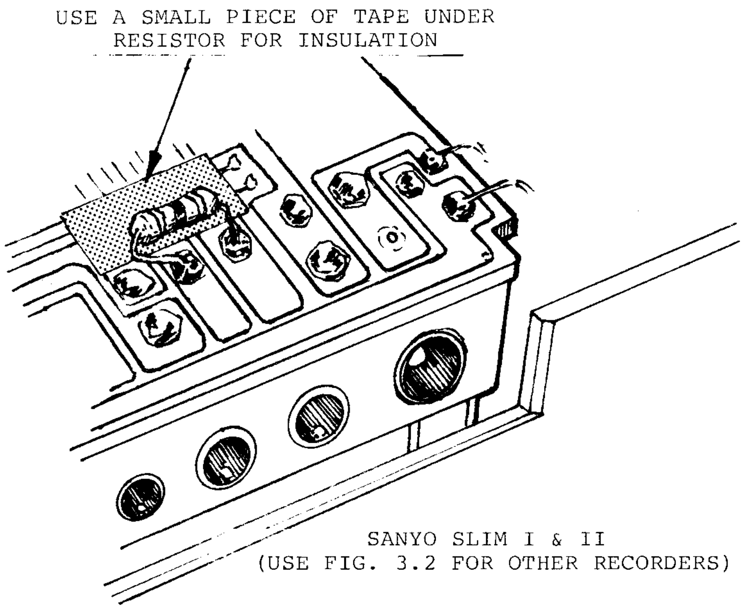
TAPE REWIND

- Place computer in READY state.
- Press cassette recorder REWIND button.
- Type VERIFY (or V shift E) then press RETURN key.
- When the cassette is rewound press the computer RUN/STOP key.
- Reset the cassette recorder REWIND button.
- We recommend that you always advance the beginning of tapes past the leader when performing SAVE operations so that you don't lose data trying to record on the leader.

OPTIONAL MODIFICATION

When the Ear plug from TAPEWORM is plugged into the Ear jack on your recorder, the speaker is shut off by a switch built into the jack. To allow for a comfortable listening level of tape data during load operations it is necessary to jumper a 1000 ohm resistor across the recorder's internal Ear/Speaker switch. This is done on the SLIM 1 and 2 by soldering the 1000 ohm resistor to the circuit board in the location shown on figure 4.5. For other recorders you can use the circuit in figure 3.2.

FIGURE 4.5



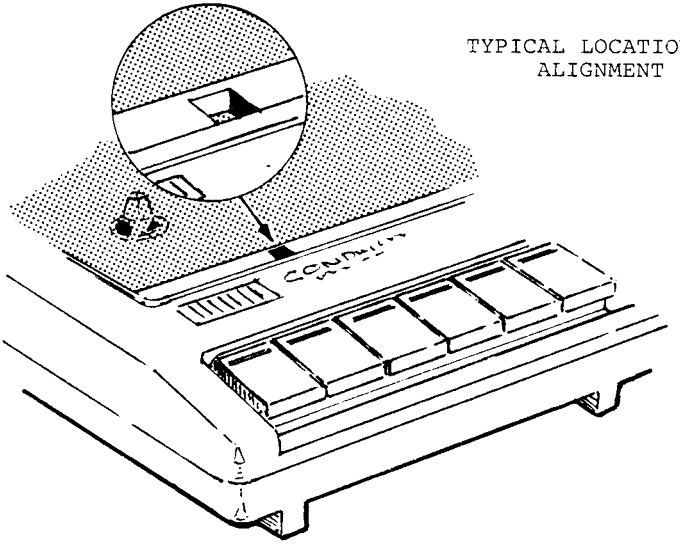
ADDITIONAL INFORMATION

HEAD ALIGNMENT

Normally all tapes recorded on your recorder will be in alignment with the tape head. However, tapes made on different recorders or some commercial tapes may not be aligned with your machine, resulting in difficult loading. The following steps are for aligning your machine.

If you are using a datasette you will need to wire the "LOAD DATA AUDIO" circuit (figure 3.2). If you have this circuit already wired, or it's equivalent, simply ignore the referance to "Ear Plug" in the following procedure.

- Unplug 'EAR' plug.
- Put in tape - Do not close cover.
- Locate alignment hole left of tape head. (see sketch)
- Set volume 1/4 to 1/2 way up - Press Play.
- Adjust screw for loudest output.
  * Don't turn far! A slight adjustment back and forth only!



TYPICAL LOCATION OF HEAD ALIGNMENT HOLE

- Other brands usually work well with TAPEWORM. However, if recorder voltage is different than 6VDC, you cannot use the power plug supplied. You can use the adapter or power source normally supplied with your recorder. This will require that you control your recorder MANUALLY since TAPE-WORM normally controls the recorder through the DC 6V plug.
- The TAPEWORM will control your 6VDC recorder for audio uses if you unplug the 'MIC' and 'EAR' plugs. You can turn the recorder <u>on</u> for playing or recording by using POKE 37148,252

FIGURE 4.6a
CLONE CIRCUITRY

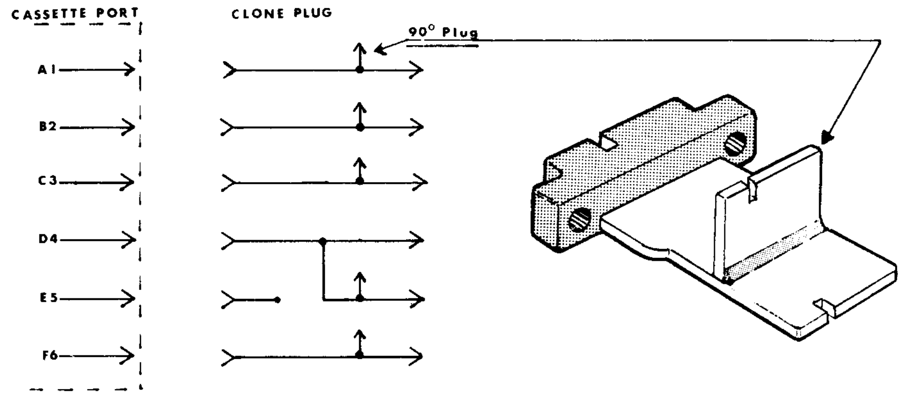


FIGURE 4.6b

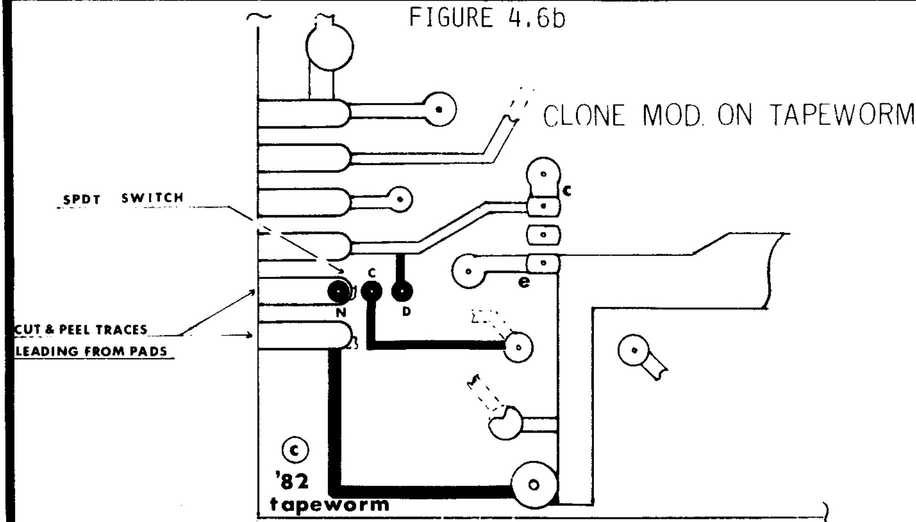


Figure 4.6a is the schematic of a clone plug. Note that if You are using one or more datasettes, that you will need the 90° plug. See figure 4.8 for PC layout.

Figure 4.6b shows how to modify a Tapeworm so that you will not need a clone plug. (You must use two standard recorders which work in the normal manner with Tapeworm to use this modification) You can either modify the artwork of figure 4.3 or remove traces from two lower pads and use jumpers for changes indicated in solid black. Clone is C-D, normal C-N.

FIGURE 4.7

CLONE HOOK-UPs

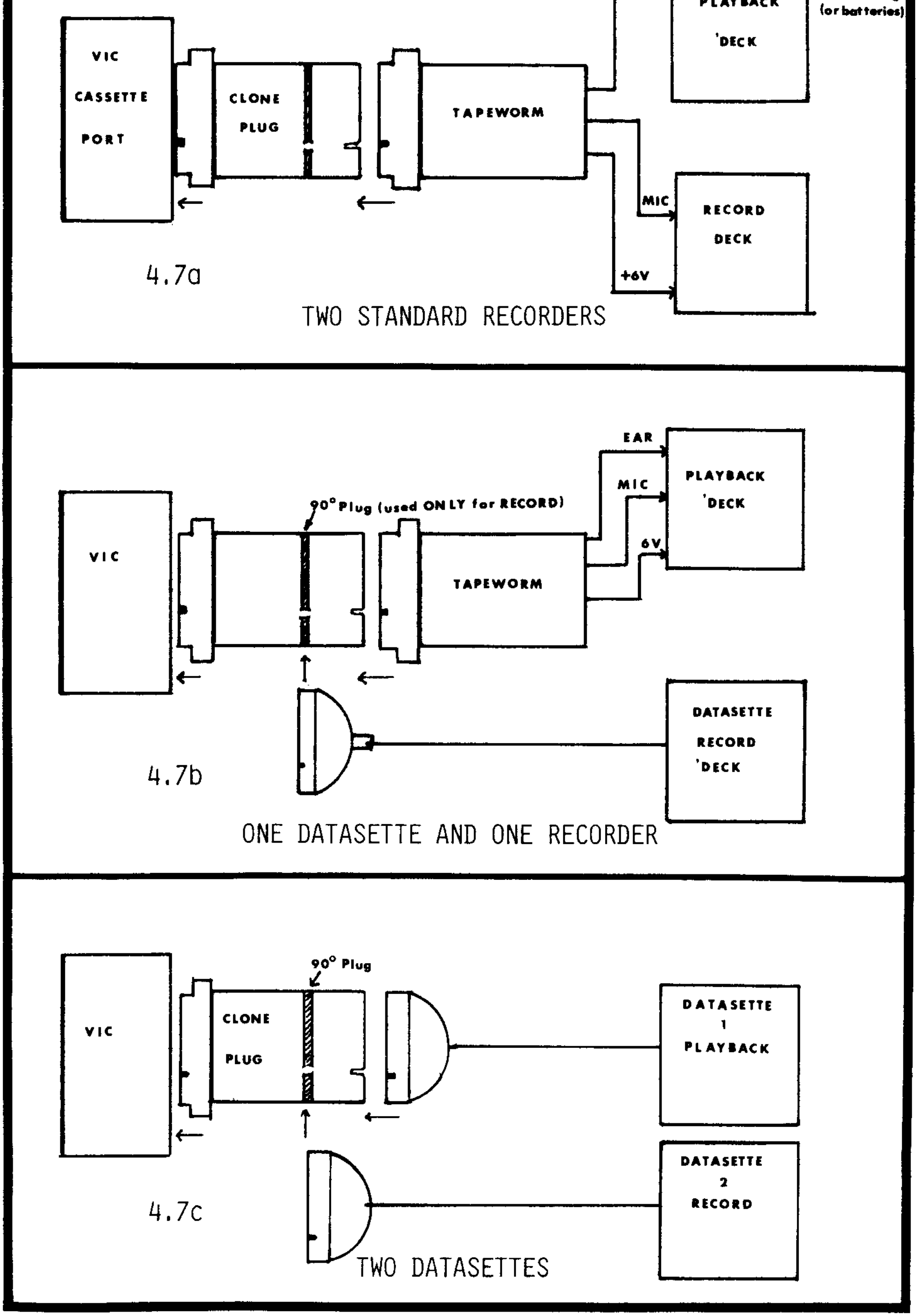
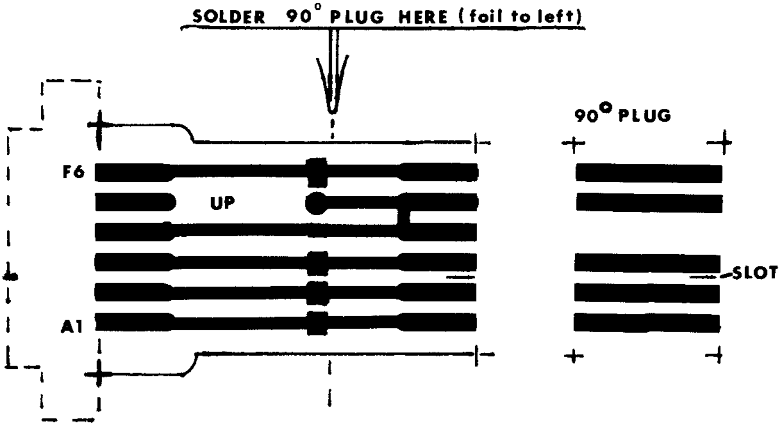


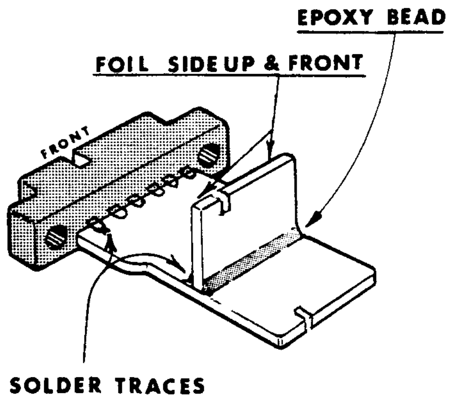
FIGURE 4.8

SUPER CLONE PLUG PC BOARD



1:1 LAYOUT

90° PLUG IS FOR RECORD ONLY (SEE FIG. 4.7)



in a program line. POKE 37148,254 turns the cassette motor off.

CLONING

Since there are so many ways to protect taped software; it is often difficult to "break the code". The Cloneplug audio duplication system provides the solution. As a dumb copier, it makes no difference what programming tricks have been used for protection.

The problem in audio duplication systems is that they are designed for audio and as such, tend to degrade digital information. Two cassette decks by themselves will rarely produce a useable copy. High quality reel to reel recorders can sometimes be used since their fidelity is somewhat better.

The most effective means turns out to be a circuit which conditions the signals like the Tapeworm does. By making a PC alteration (fig. 4.6b) to Tapeworm you can use it for this purpose with two audio recorders. Figure 4.6.a and 4.7a-c show a simple plug which can be used if one or both recorders are datasettes. Before attempting cloning be sure that the recorders you are using are compatible with the Tapeworm and VIC. Study 4.7a-c carefully to determine which combination suits the recorders you are using.

CLONING PROCEDURE

1. Determine correct hookup from figure 4.7 based on the type of recorders you are using.
2. Place original tape in PLAYBACK DECK.
3. Place blank tape in RECORD DECK.
4. Make sure proper switches are pressed. Enter LOAD command.
5. When done you can verify cloned copy by loading it.
6. Best results are obtained by using the same deck to load as was used to record the clone.

Remember that cloned copies exhibit some degradation from the original. Cloning clones will only work to about three or four levels. Chapter five deals with methods of using the computer to save copies which turn out equal to the original in quality.

CHAPTER FIVE

TAPE CONQUEST

TAPE COPY PROCEDURES

This chapter deals with two methods for saving taped programs directly from the VIC. When possible, this is preferable to cloning since the copy comes directly from the computer. The digital quality of the tape is thus identical to the original. The first method uses SAVMACH (tm) software which is listed at the end of this chapter. For machine language programs, Savmach will eliminate the need to use editor-assemblers. The second method involves a complex but effective method we have dubbed "Label Switching".

SAVMACH

Chapter three described the process of using the tape buffer for locating machine programs. Normally once this has been done you would use VIC MON to save the program. Savmach does all this for you. By using a two-step process, Savmach first reads the buffer immediately after the leader load (thus circumventing any chance that the program load would destroy the buffer). Secondly, Savmach performs a machine save. All of this is done by using two SYS numbers. Savmach resides in block 5 so that you can use it on machine programs which reside in any of the RAM locations. This provides another advantage over VIC MON which will not work for block 3 programs.

SAVMACH PROCEDURE

1. Set RAM to block 1, Write Enable ON.
2. Load Savmach using LOAD"",1,1
3. Switch RAM block 1 OFF and block 5 ON.
4. Type NEW and hit RETURN.
5. Load header only. Do this by typing LOAD"",1,1 and waiting for FOUND message then hitting RUN/STOP key.
6. Type SYS41168 and hit RETURN. This stores original header in memory.
7. Rewind tape and load entire program. (Be sure to have computer memory configured the way the program normally requires with the exception that block 5 and write enable for that block must be left ON!)
8. Insert blank tape in recorder and set for recording.
9. Type SYS41200 ("Saving" message will appear).
10. The copied program tape should load and run in the normal manner. NOTE: If original program normally changes screen content while loading, you will have to clear the screen and type in the information that the original program put there. Do this after loading and before running your copy. Cloning or label switching would be a better choice on such programs.

LABEL SWITCHING

Label switching is a virtually foolproof method of copying taped programs. It requires the use of block 1 for

8K or blocks 1 and 2 for programs up to 16K. Thus its primary use is for 16K or shorter programs which do not exceed block 2 boundaries.

Label switching for tapes is akin to block switching for cartridges. The general concept is to fool the computer into loading the taped program into block 1 (or 1&2) where it can't self destruct. The greatest advantage to label switching is that no known protection method will affect the copy. It is therefore more reliable than Savmach and it will produce better copies than cloning. It is also the most difficult to master. Label switching requires the addition of a "Save Data" audio modification as shown in figure 3.2. Since this is one of the few times you will use the save audio modification, you may wish to use a 1K ohm resistor in series with an earphone and simply "clip" it across cassette write, and ground (E5-A1).

If you are using a datasette, you will need to wire a jumper across the A-1 and F-6 wires from the Cassette Port on the VIC. The Tapeworm and some other interfaces already have the cassette switch wired "closed". You will also need to make a tape of HDRSWAP which is listed at the end of the chapter.

LABEL SWITCHING PROCEDURE

1. LOAD "HDRSWAP",1,1

2. Place 8K expander RAM inblock 1 (or 1&2) with Write enable ON.

3. Put original tape to be copied in recorder. Type LOAD"",1,1 and hit RETURN key.

4. Hit RUN/STOP IMMEDIATELY after FOUND message appears. Don't rewind tape! Insert blank tape and prepare for recording.

5. Type SYS4816 (RETURN) Press RECORD/PLAY. Type SYS 4884 (RETURN).

6. Hit RUN/STOP approximately 1 second after header ends. You must use the "Save Data" audio circuit to listen to and stop the saving process. Avoid recording program data! This is the "DUMMY HEADER".

7. Rewind Dummy header then load it using LOAD"",1,1 (RETURN).

8. Stop RECORDER (do not use RUN/STOP key) about one second after FOUND message appears.

9. Insert original tape and push PLAY. Wait for READY message. (The original program is now in block 1 (or 1 and 2).

10. Turn OFF write enable for blocks used.

11. Insert a blank tape and set recorder to RECORD. (You may use same tape as dummy header was on as you are now done with it).

12. Type SYS4848 (RETURN).

13. Listen to save data, wait about 1 second after header, then hit RUN/STOP key.

14. Push STOP on recorder.

15. Type SYS4884 (RETURN)

16. Immediatly after the header, push RECORD/PLAY on recorder.

17. This procedure will place and End Of File (EOF) message at the end of the program. You can eliminate this by pushing RUN/STOP at the end of the program when you hear the EOF tone leader.

18. To make another copy, repeat steps 11-16.

19. This should result in a good copy. If so, you may apply for buccaneer rating!

ASSEMBLY

```
20D0 LDX #$00
20D2 LDA $033C,X
20D5 STA $A000,X
20D8 INX
20D9 CPX #$BF
20DB BEQ $20E0
20DD JMP $A0D2
20E0 LDA $A001
20E3 STA $FB
20E5 LDA $A002
20E8 STA $FC
20EA NOP
20EB NOP
20EC NOP
20ED NOP
20EE NOP
20EF RTS
20F0 LDA #$01
20F2 LDX #$01
20F4 LDY #$FF
20F6 NOP
20F7 JSR $FFBA
20FA LDA #$10
20FC LDX #$05
20FE LDY #$A0
2100 JSR $FFBD
2103 LDA #$FB
2105 LDX $A003
2108 LDY $A004
210B JSR $FFD8
210E NOP
210F NOP
2110 NOP
2111 NOP
2112 NOP
2113 NOP
2114 NOP
2115 RTS
2116 BRK
```

MACHINE CODE

```
.:20D0 A2 00 BD 3C 03
.:20D5 9D 00 A0 E8 E0
.:20DA BF F0 03 4C D2
.:20DF A0 AD 01 A0 85
.:20E4 FB AD 02 A0 85
.:20E9 FC EA EA EA EA
.:20EE EA 60 A9 01 A2
.:20F3 01 A0 FF EA 20
.:20F8 BA FF A9 10 A2
.:20FD 05 A0 A0 20 BD
.:2102 FF A9 FB AE 03
.:2107 A0 AC 04 A0 20
.:210C D8 FF EA EA EA
.:2111 EA EA EA EA 60
.:2116 00 ED 40 FF 20
```

```
SYS 41168
SYS 41200
```

THE MACHINE LANGUAGE PROGRAMS CAN BE ENTERED BY USING AN EDITOR ASSEMBLER SUCH AS VICMON OR BY ENTERING THE HEX CODES WITH A MACHINE CODE LOADER.

HDRSWAP

```
12D0 LDX #$00
12D2 LDA $033C,X
12D5 STA $1200,X
12D8 INX
12D9 CPX #$BF
12DB BEQ $12E0
12DD JMP $12D2
12E0 LDA $1201
12E3 STA $FB
12E5 LDA $1202
12E8 STA $FC
12EA NOP
12EB NOP
12EC NOP
12ED NOP
12EE NOP
12EF RTS
12F0 LDA #$01
12F2 LDX #$01
12F4 LDY #$FF
12F6 NOP
12F7 JSR $FFBA
12FA LDA #$10
12FC LDX #$05
12FE LDY #$12

1300 JSR $FFBD
1303 LDA #$FB
1305 LDX $1203
1308 LDY $1204
130B JSR $FFD8
130E RTS
130F NOP
1310 BRK
1311 BRK
1312 NOP
1313 NOP
1314 CLC
1315 LDA $1203
1318 STA $1310
131B LDA $1204
131E ADC #$10
1320 STA $1311
1323 CLC
1324 LDA $FB
1326 STA $FD
1328 CLC
1329 LDA $FC
132B ADC #$10
132D STA $FE
132F CLC
1330 LDA #$01
1332 LDX #$01
1334 LDY #$FF
1336 NOP
1337 JSR $FFBA
133A LDA #$10
133C LDX #$05
133E LDY #$12
1340 JSR $FFBD
1343 LDA #$FD
1345 LDX $1310
1348 LDY $1311
134B JSR $FFD8
134E RTS
134F BRK
```

SYS 4816

SYS 4848

SYS 4884

CHAPTER SIX

ISLE OF ROM

CARTRIDGE COPYING METHODS

Cartridges are quite easy to save if they are switched into unprotected block locations then saved as machine language programs. This procedure, as covered in this chapter, gets around the commonly used protection schemes for cartridges. In this manner the cartridges can be put on tape or disk and used simply by loading them into RAM. We call these tapes "cartridge tapes". Using the tapes or disks and RAM, you will be able to load new cartridges without shutting off the computer and plugging-in new cartridges each time. This will save a great deal of wear and tear on your edge connectors and eliminate the need for costly expansion bus systems. Furthermore, you can have multiple programs in RAM (one in each 8K) and switch back and forth with the block switches.

The process of running cartridge tapes involves loading them into RAM which can be switched into the location the cartridge normally resides in. As discussed in earlier chapters, we usually need to protect the RAM from being written to. (Write enable switch, figures 3.1 and 6.1) with the write enable switch OFF, the RAM works identically to the ROM it is substituting for with the exception that power must be maintained to the RAM.

This chapter presents the ROMULATOR (tm) system which provides hardware, software, and procedures to make cartridge tapes and disks. The system requires 8K RAM for 8K cartridges and 16K RAM for 16K cartridges. Since many cartridges are 8K you may choose to start by doing only 8Ks and deferring the expense of a 16K (or two 8Ks) until later. If so, simply ignore the information given for the 16K procedures.

The ROMULATOR program is machine language and it's listing is at the end of this chapter. We have included both the assembly listing and the machine code listing for your convienience. Appendix B lists availability of software and hardware from PSIDAC if you choose not to "roll your own". Also listed at the end of the chapter is BLOCK CHECK which can help you discover the loacations of ROM in cartridges.

ROMULATOR

FEATURES

* Makes tape or disk duplicates of VIC-20 cartridge software.
* Provides "Block Check" to find locations of ROM/RAM.
* Allows 8K RAM expander to be located in any block.
* Provides security of having backups of your software.
* Provides rapid access of software by allowing up to four programs to be in RAM & unlimited disk libraries.
* No need to shut off computer to use different programs.
* Simple to use!

DESCRIPTION

The ROMULATOR system consists of two circuit cards and two programs. This system allows RAM & ROM memory to be interrogated, relocated and saved to TAPE or DISK. In this manner you can make backup copies of ROM cartridges. The RAM SWITCH card will allow you to easily switch the block location of your 8K RAM expander. 16K cartridges will require two switchable 8K blocks. Your RAM can occupy any available block including 5 which is the cartridge block.

As with any recording duplicating system, ROMULATOR should be used only for making backup copies for your own private use. It should be used only in accordance with existing copyright law.

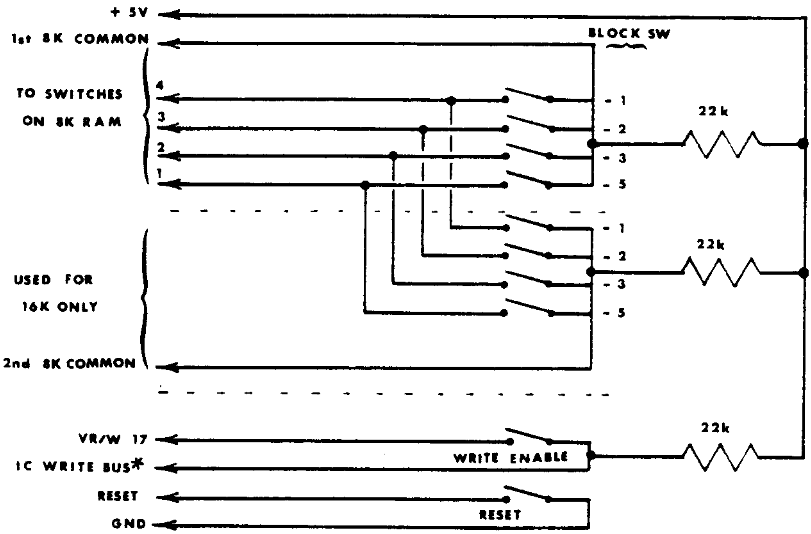
8K RAM SWITCH MODIFICATION

If you own an 8K RAM made by Commodore, and you don't have an expansion bus or other way to switch RAM locations, you will need to install the RAM SWITCH card. This will allow you to switch the RAM to any block while power is on.

The RAM SWITCH features a memory write enable switch which can be used to cause the RAM to look like ROM to the computer. There is also a reset switch which restarts the computer without losing RAM data!

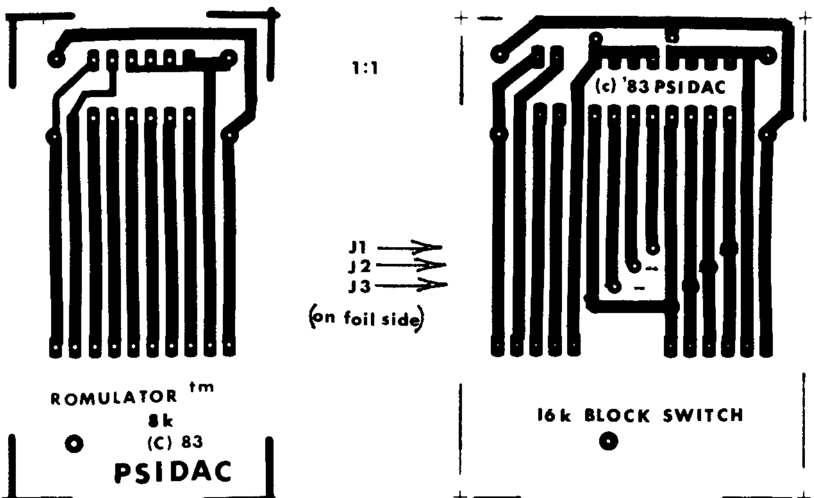
FIGURE 6.1

RAM BLOCK SWITCH (8 or 16K)



*SEE FIGURE 3.1 & 6.4

FIGURE 6.2
PC CARD LAYOUT FOR COMMODORE RAMs



ALL SWITCHES AND JUMPERS ON RAM CARD MUST BE OPEN OR OFF !

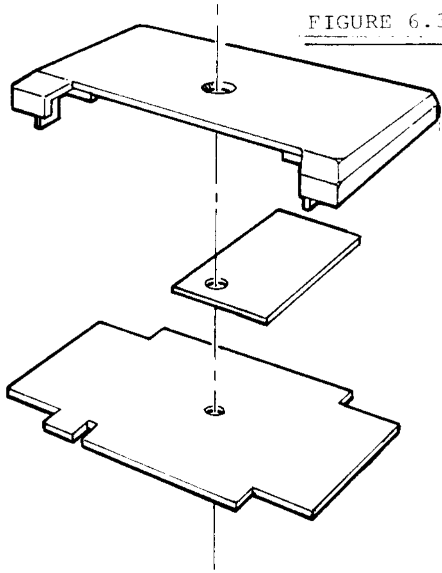
Figure 6.1 shows the schematics for an 8K RAM switch and a 16K RAM switch. Figure 6.2 gives the corresponding PC card layouts for Commodore RAMs. If you have two 8K expanders, each will need a RAM switch to be used with 16K cartridge tapes. If you own other brands of RAM expanders, you will need to devise a block switch, and write enable switch scheme from the information given. Most of the commercial memories have an easily accessible block switch already. You may wish to wire the RESET switch on the computer instead of on the RAM. Although the SYS64802 reset usually works, you will find a hardware reset very convenient.

RAM SWITCH INSTALLATION

FOR COMMODORE RAMS

1. Disassemble 8K case (1 screw, 4 snaps)
2. Orient RAM SWITCH as shown. (fig. 6.3.)

FIGURE 6.3



3. Tape card to 8K with holes centered.

4. Tack solder wires as shown in (fig. 6.4.) (1-1,2-2,...)

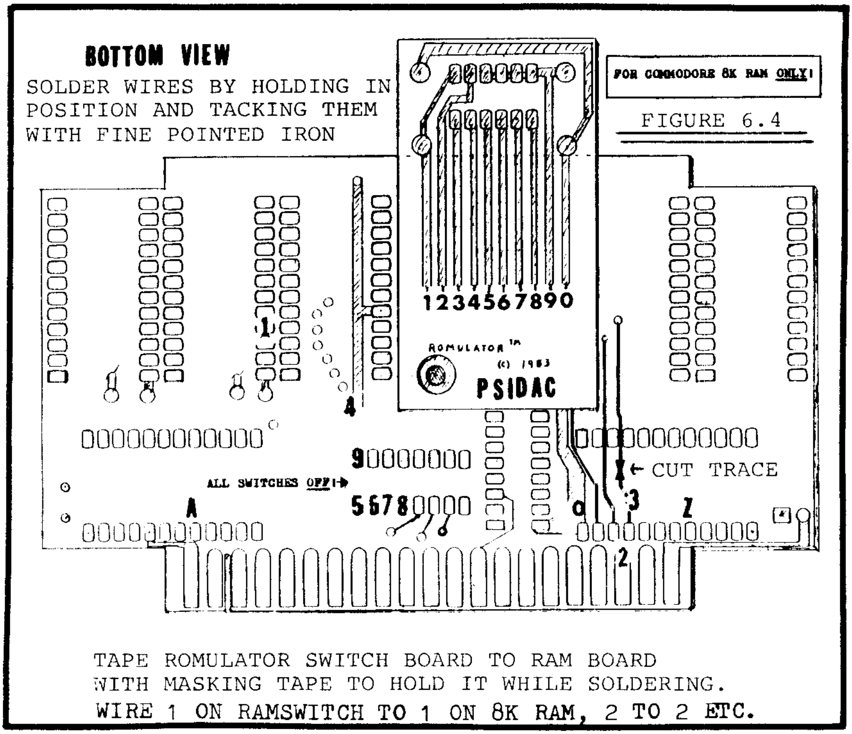


FIGURE 6.4

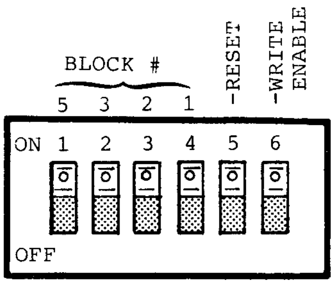
5. Cut trace indicated by X.

6. Study diagram carefully to be sure job is CORRECT!

7. * ALL SWITCHES OF 8K BOARD MUST BE OFF!*

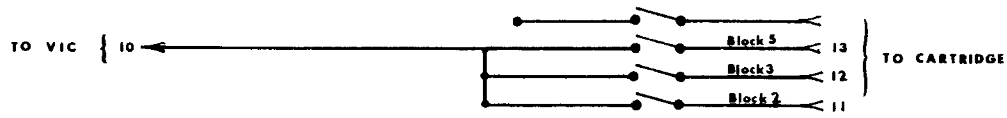
8. Reassemble case. (If properly installed, RAM SWITCH will be held securely by 8K case & screw).

Your New Block Select Switches Are:



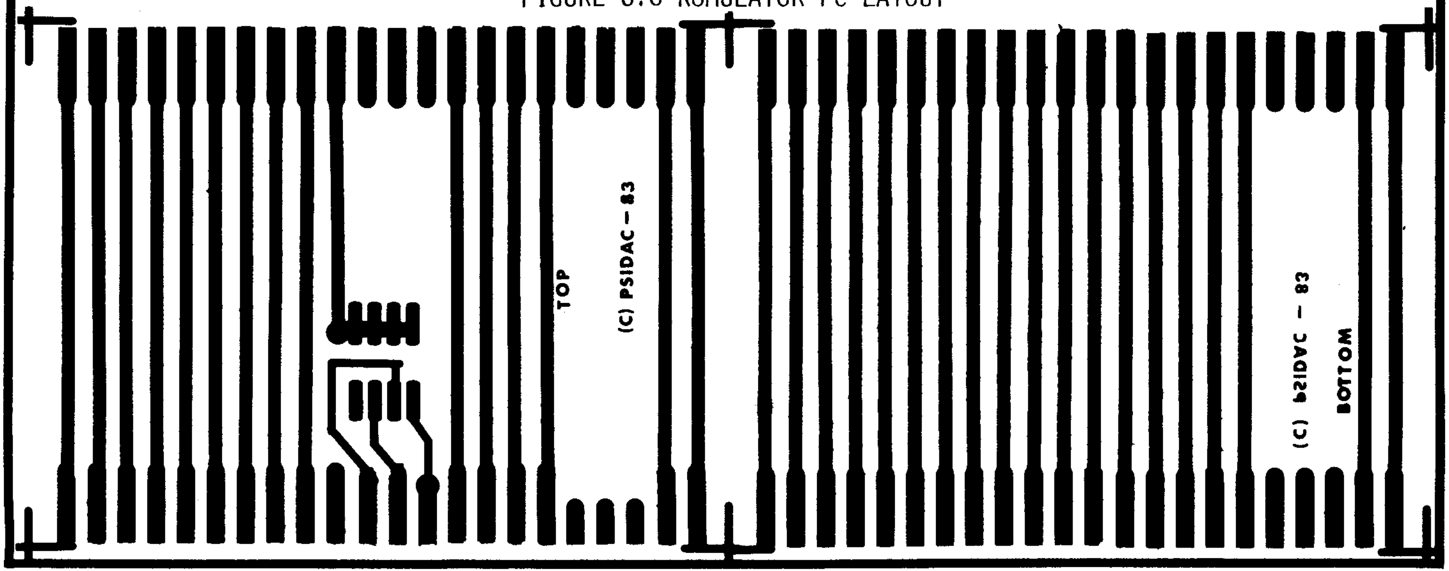
*All Switches Shown In "ON" Position

FIGURE 6.5
ROMULATOR SCHEMATIC



Only the modified expansion bus wiring is shown. Romulator doesn't affect the rest of the bus. The Romulator board allows any cartridge to be switched to block one where it won't auto-start, and from where the Kernal allows a SAVE. The PC layout below is for a two-sided board. Note top and bottom side drawings.

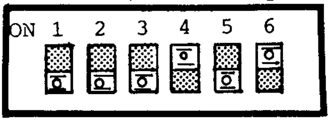
FIGURE 6.6 ROMULATOR PC LAYOUT



TESTING 8K RAM

After modifying 8K RAM perform the following tests.

1. Install in computer with power off.
2. Set to block 1 RAM, turn on power.



"BLOCK ONE"
BLOCK SWITCH
SETTING

3. 11775 bytes free should be displayed. If not shut off power and locate problem then repeat 1-4.
4. Load "Block Check" and run.
5. Block Check will tell you which block the RAM is in or if write enable is OFF, it will identify it as ROM.
6. Run it once for each setting of the block switches both with the protect switch ON & OFF. (8 tests) Each time it should agree with the switch settings.

SAVING 8K CARTRIDGES
TO TAPE OR DISK

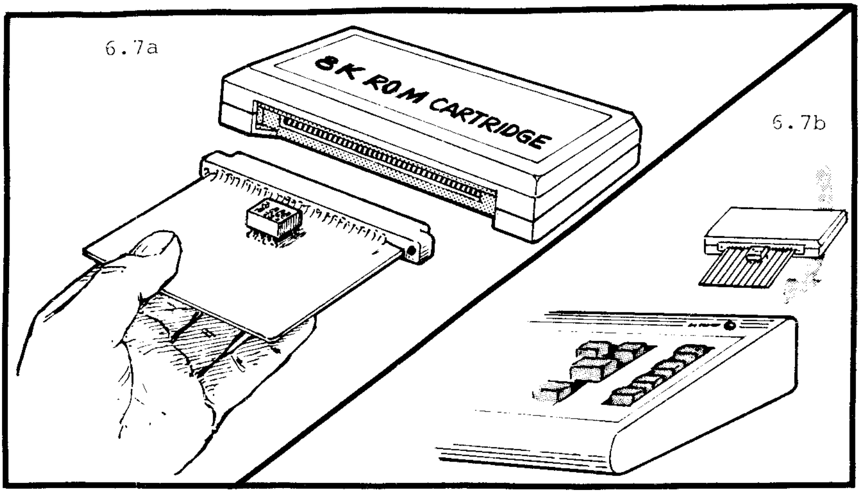
Materials Required

1. Romulator card (figures 6.5, & 6.6)
2. Romulator program
3. 8K or 16K switchable RAM
4. 8K or 16K ROM (to be copied)
5. VIC-20 with monitor, datasette or disk

Procedure

1. Turn off computer
2. Plug ROM cartridge into Romulator card (fig. 6.7.a)

3. Plug the Romulator card into the expansion interface slot (fig. 6.7.b)



4. Set Romulator dip switches in their correct positions for current block being saved.

5. Note that most 8K cartridges are block 5*, most 16K are 5 and 3. For 16K cartridges, only save ONE block at a time. That is, perform two separate saves.

6. Block one ROM does not require use of the ROMULATOR PC card.

TABLE 6.1

Block To Save	Switch Positions 1	2	3	4
2	OFF	ON	OFF	OFF
3	OFF	OFF	ON	OFF
5*	ON	OFF	OFF	OFF

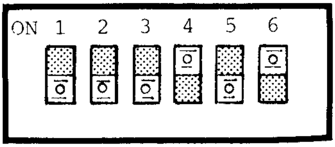
7. If you are unsure of which block to select, refer to BLOCK CHECK for further information.
8. Turn on computer!

   If the monitor fails to display the "3583 BYTES FREE" message within 3 seconds then turn off the computer and locate the problem. Double check procedure 1-7.
9. Load Romulator program with LOAD"",1,1 command.
10. If the program fails to load see "Head Alignment" procedure.
11. Install a blank tape in the datasette and prepare for recording. *For disk, select properly formatted disk and insert in disk drive.
12. Type in the command SYS 6400 and hit RETURN.
13. The "ROMULATOR" heading should now be displayed on the monitor.
14. For block number, type in the appropriate number and hit return. *For most cartridges this is 5.
15. For device number type in a 1 for tape or an 8 for disk.
16. For name type in a 15 letter or less name. *No quotes are needed for name.
17. Make sure blank tape is ready for recording. Hit RETURN and allow 3 minutes for recording. *Allow about 20 seconds for disk.
18. If another copy is needed repeat process from step 12.
19. For second half of 16K, reset ROMULATOR card dip switches according to table 6.1 and the second block location you are saving. Then repeat the procedure from step 10.
20. To verify recording, see "Loading 'Cartridge Tapes'".

LOADING 'CARTRIDGE TAPES'

OR DISKS

1. Make sure switchable 8K expander is installed. (or 16K for 16K cartridge tapes).
2. Set block one (4) ON and write enable (6) ON, for only one 8k RAM. The other 8K, if being used, should have all switches OFF!



3. Turn on computer (if already on push RESET momentarily ON or type SYS 64802 and hit RETURN)
4. The monitor should now display "11775 BYTES FREE" (if not, check switches, proper insertion, reset switch etc. & repeat 1-4).
5. Select tape and perform LOAD"",1,1 allow 3 minutes for tape load. *For disk, type LOAD"Prg Name*",8,1
6. When VIC comes back with READY message; switch OFF Memory Protect and Block One (switches 6 & 4). For 16K cartridge tape, you will need to perform steps 2-6 for the other 8K RAM. DON'T TURN OFF COMPUTER!
7. Turn ON block(s) which correspond to the normal location of the program. Usually this is block 5, or 5 and 3 for 16K RAMs. With 16K RAMs be sure not to confuse which 8K section contains which half of the program!

*Push Reset momentarily ON or type SYS 64802 RETURN. The ROM program should begin.

8. If program does not operate, repeat these steps. Also verify proper head alignment of recorder etc.

* Note that best results are obtained if the same recorder is used to SAVE and LOAD programs. It is illegal to make copies of copyrighted software for someone else.

BLOCK CHECK
FOR AUTO START CARTRIDGES

Use this method to determine block locations and contents of unknown auto start cartridges.

If program starts when VIC is turned on, cartridge is in block 5. It may also contain other memory which can be located by steps 1-4.

1. Install cartridge in Romulator card and VIC (fig. 6.7a & 6.7b)
2. All dip switches should be OFF. Turn on VIC & LOAD "BLOCK CHECK" (do not RUN yet)
3. Turn ON first dip switch & RUN. (Do this one at a time as table 6.2 shows)
4. Determine block location by table 6.2 and message

   *For this test monitor will always display BLOCK 1.

TABLE 6.2

SWITCH ON	"RAM/ROM IN BLOCK 1" MESSAGE MEANS:
1	RAM/ROM IN BLOCK 5
2	RAM/ROM IN BLOCK 2
3	RAM/ROM IN BLOCK 3
4	INVALID SWITCH POS.

NON AUTO START CARTRIDGES

Many cartridges require SYS numbers to operate. You can determine the block these start in by comparing the SYS number to the memory map (figure 2.1). A SYS24577 for example, starts in block three. You should use BLOCKCHECK to look for other possible ROM locations. If a cartridge contains ROM in more than one block, you should save each block located using the ROMULATOR cartridge save procedure.

--Non auto-start cartridges can be examined by simply plugging them into the VIC and running BLOCKCHECK.--

HEAD ALIGNMENT

If you experience difficulty with tape loading you probably need to align your heads or use better tape. If you own the Commodore recorder you will need some way to hear the load data. (see fig. 3.2)

For other recorders see chapter 4 "HEAD ALIGNMENT PROCEDURE".

```
1900 LDY #$00
1902 JSR $1B00
1905 LDY #$03
1907 LDX #$08
1909 CLC
190A JSR $FFF0
190D CLC
190E JSR $FFCF
1911 STA $18E0
1914 JSR $FFCF
1917 CMP #$0D
1919 BNE $1914
191B CLC
191C LDA $18E0
191F CMP #$31
1921 BEQ $1935
1923 CMP #$32
1925 BEQ $1935
1927 CMP #$33
1929 BEQ $1935
192B CMP #$35
192D BEQ $1935
192F JSR $1B10
1932 JMP $1900
1935 JSR $1B38
1938 NOP
1939 NOP
193A LDX #$0C
193C LDY #$03
193E CLC
193F JSR $FFF0
1942 CLC
1943 JSR $FFCF
1946 STA $18E1
1949 JSR $FFCF
194C CMP #$0D
194E BNE $1949
1950 CLC
1951 LDA $18E1
1954 CMP #$31
1956 BEQ $1966
1958 CMP #$38
195A BEQ $1966
195C CMP #$39
195E BEQ $1966
1960 JSR $1B10
1963 JMP $1900
1966 JSR $1B38
1969 LDX #$10
196B LDY #$03
196D CLC
196E JSR $FFF0
1971 CLC
1972 LDX #$00
1974 NOP
1975 JSR $FFCF
1978 STA $18E2,X
197B CPX #$10
197D BEQ $1980
197F INX
1980 CMP #$0D
1982 BNE $1975
1984 DEX
1985 LDA #$20
1987 CPX #$11
1989 BEQ $1992
198B STA $18E2,X
198E INX
198F JMP $1987
1992 NOP
1993 NOP
1994 LDX $18E0
1997 LDA #$30
1999 STA $18E0
199C TXA
199D SBC $18E0
19A0 STA $18E0
19A3 LDX $18E1
19A6 LDA #$30
19A8 STA $18E1
19AB TXA
19AC SBC $18E1
19AF STA $18E1
19B2 CLC
19B3 NOP
19B4 NOP
19B5 NOP
19B6 LDA #$01
19B8 LDX $18E1
19BB LDY #$FF
19BD JSR $FFBA
19C0 LDA #$10
19C2 LDX #$E2
19C4 LDY #$18
19C6 JSR $FFBD
19C9 JSR $1B38
19CC LDA $18E0
19CF CMP #$01
19D1 BNE $19DC
19D3 BNE $19DC
19D5 LDA #$01
19D7 STA $9800
19DA JMP $19F9
19DD CMP #$02
19DF BNE $19E8
19E1 LDA #$90
19E3 STA $9800
19E6 JMP $19F9
19E9 CMP #$03
19EB BNE $19F4
19ED LDA #$A0
19EF STA $9800
19F2 JMP $19F9
```

(cont ...)

```
19F5 LDA #$C0
19F7 STA $9800
19FA CLC
19FB LDY #$03
19FD LDX #$11
19FF JSR $FFF0
1A02 LDX #$00
1A04 STX $FB
1A06 LDX #$20
1A08 STX $FC
1A0A LDX #$00
1A0C LDY #$40
1A0E NOP
1A0F NOP
1A10 LDA #$FB
1A12 JSR $FFD8
1A15 BCC $1A40
1A17 CMP #$05
1A19 BNE $1A38
1A1B LDY #$B2
1A1D JSR $1B00
1A20 JSR $1B10
1A23 JSR $1B60
1A26 JSR $1B10
1A29 JSR $1B60
1A2C JSR $1B10
1A2F JSR $1B60
1A32 JSR $1B10
1A35 JSR $1B60
1A38 RTS
1A39 NOP
1A3A NOP
1A3B NOP
1A3C NOP
1A3D NOP
1A3E NOP
1A3F NOP
1A40 JSR $1B38
1A43 JSR $1B60
1A46 JSR $1B38
1A49 JSR $1B60
1A4C JSR $1B38
1A4F JSR $1B60
1A52 JSR $1B38
1A55 RTS
1A56 BRK
1A57 BRK
 |     "
 |     "
 ~     "
 |     "
 v
1AFF   "

1B00 LDA $1800,Y
1B03 CMP #$04
1B05 BEQ $1B0E
1B07 JSR $FFD2
1B0A INY
1B0B JMP $1B00
1B0E RTS
1B0F BRK
1B10 LDA #$0F
1B12 STA $900E
1B15 LDA #$C0
1B17 STA $900D
1B1A LDA #$00
1B1C LDX #$00
1B1E LDY #$00
1B20 JSR $FFDB
1B23 NOP
1B24 JSR $FFDE
1B27 CMP #$15
1B29 BNE $1B24
1B2B LDA #$00
1B2D STA $900E
1B30 LDA #$00
1B32 STA $900D
1B35 RTS
1B36 NOP
1B37 NOP
1B38 LDA #$0F
1B3A STA $900E
1B3D LDA #$DC
1B3F STA $900C
1B42 LDA #$00
1B44 LDX #$00
1B46 LDY #$00
1B48 JSR $FFDB
1B4B NOP
1B4C JSR $FFDE
1B4F CMP #$08
1B51 BNE $1B4C
1B53 LDA #$00
1B55 STA $900E
1B58 LDA #$00
1B5A STA $900C
1B5D RTS
1B5E NOP
1B5F NOP
1B60 LDA #$00
1B62 LDX #$00
1B64 LDY #$00
1B66 JSR $FFDB
1B69 NOP
1B6A JSR $FFDE
1B6D CMP #$02
1B6F BNE $1B6A
1B71 RTS
1B72 BRK
```

Type in using editor assembler such as Vicmon or use a machine loader routine and enter machine code listing. If typing assembly language, be sure to include machine data table $1800-$18FF.

ROMULATOR MACHINE CODE

(Data table $1800-$18FF must be entered)

DATA

```
.:1800 93 20 20 90 72
.:1805 72 72 72 72 72
.:180A 72 72 72 72 72
.:180F 72 72 72 72 72
.:1814 72 72 0D 20 20
.:1819 1F 52 4F 4D 55
.:181E 4C 41 54 4F 52
.:1823 20 53 41 56 45
.:1828 20 4D 4F 4E 0D
.:182D 20 20 90 A3 A3
.:1832 A3 A3 A3 A3 A3
.:1837 A3 A3 A3 A3 A3
.:183C A3 A3 A3 A3 A3
.:1841 A3 0D 20 20 20
.:1846 20 1C 28 43 29
.:184B 1E 50 53 49 44
.:1850 41 43 20 1C 31
.:1855 39 38 33 0D 00
.:185A 11 11 1F 2A 45
.:185F 4E 54 45 52 20
.:1864 42 4C 4F 43 4B
.:1869 20 54 4F 20 53
.:186E 41 56 45 0D 11
.:1873 20 31 2E 00 0D
.:1878 11 1C 2A 45 4E
.:187D 54 45 52 20 53
.:1882 41 56 49 4E 47
.:1887 20 44 45 56 49
.:188C 43 45 0D 11 20
.:1891 32 2E 00 0D 11
.:1896 1E 2A 45 4E 54
.:189B 45 52 20 50 52
.:18A0 4F 47 52 41 4D
.:18A5 20 4E 41 4D 45
.:18AA 0D 11 20 33 2E
.:18AF 1F 0D 04 0D 11
.:18B4 9F 2A 2A 43 48
.:18B9 45 43 4B 20 44
.:18BE 49 53 43 20 53
.:18C3 59 53 54 45 4D
.:18C8 00 2A 2A 1F 0D
.:18CD 04 00 00 00 00
.:18D2 00 00 00 00 00
.:18D7 00 00 00 00 00
.:18DC 00 00 00 00 01
.:18E1 01 31 20 20 20
.:18E6 20 20 20 20 20
.:18EB 20 20 20 20 20
.:18F0 20 20 20 00 00
.:18F5 00 00 00 00 00
.:18FA 00 00 00 00 00
.:18FF 00 A0 00 20 00
```

PROGRAM

```
.:1900 A0 00 20 00 1B
.:1905 A0 03 A2 08 18
.:190A 20 F0 FF 18 20
.:190F CF FF 8D E0 18
.:1914 20 CF FF C9 0D
.:1919 D0 F9 18 AD E0
.:191E 18 C9 31 F0 12
.:1923 C9 32 F0 0E C9
.:1928 33 F0 0A C9 35
.:192D F0 06 20 10 1B
.:1932 4C 00 19 20 38
.:1937 1B EA EA A2 0C
.:193C A0 03 18 20 F0
.:1941 FF 18 20 CF FF
.:1946 8D E1 18 20 CF
.:194B FF C9 0D D0 F9
.:1950 18 AD E1 18 C9
.:1955 31 F0 0E C9 38
.:195A F0 0A C9 39 F0
.:195F 06 20 10 1B 4C
.:1964 00 19 20 38 1B
.:1969 A2 10 A0 03 18
.:196E 20 F0 FF 18 A2
.:1973 00 EA 20 CF FF
.:1978 9D E2 18 E0 10
.:197D F0 01 E8 C9 0D
.:1982 D0 F1 CA A9 20
.:1987 E0 11 F0 07 9D
.:198C E2 18 E8 4C 87
.:1991 19 EA EA AE E0
.:1996 18 A9 30 8D E0
.:199B 18 8A ED E0 18
.:19A0 8D E0 18 AE E1
.:19A5 18 A9 30 8D E1
.:19AA 18 8A ED E1 18
.:19AF 8D E1 18 18 EA
.:19B4 EA EA A9 01 AE
.:19B9 E1 18 A0 FF 20
.:19BE BA FF A9 10 A2
.:19C3 E2 A0 18 20 BD
.:19C8 FF 20 38 1B AD
.:19CD E0 18 C9 01 D0
.:19D2 09 D0 07 A9 01
.:19D7 8D 00 98 4C F9
.:19DC 19 C9 02 D0 07
.:19E1 A9 90 8D 00 98
.:19E6 4C F9 19 C9 03
.:19EB D0 07 A9 A0 8D
.:19F0 00 98 4C F9 19
.:19F5 A9 C0 8D 00 98
.:19FA 18 A0 03 A2 11
.:19FF 20 F0 FF A2 00
```

```
.:1A04 86 FB A2 20 86
.:1A09 FC A2 00 A0 40
.:1A0E EA EA A9 FB 20
.:1A13 D8 FF 90 29 C9
.:1A18 05 D0 1D A0 B2
.:1A1D 20 00 1B 20 10
.:1A22 1B 20 60 1B 20
.:1A27 10 1B 20 60 1B
.:1A2C 20 10 1B 20 60
.:1A31 1B 20 10 1B 20
.:1A36 60 1B 60 EA EA
.:1A3B EA EA EA EA EA
.:1A40 20 38 1B 20 60
.:1A45 1B 20 38 1B 20
.:1A4A 60 1B 20 38 1B
.:1A4F 20 60 1B 20 38
.:1A54 1B 60 00 00 00
```

(THIS MEMORY NOT USED)

```
.:1B00 B9 00 18 C9 04
.:1B05 F0 07 20 D2 FF
.:1B0A C8 4C 00 1B 60
.:1B0F 00 A9 0F 8D 0E
.:1B14 90 A9 C0 8D 0D
.:1B19 90 A9 00 A2 00
.:1B1E A0 00 20 DB FF
.:1B23 EA 20 DE FF C9
.:1B28 15 D0 F9 A9 00
.:1B2D 8D 0E 90 A9 00
.:1B32 8D 0D 90 60 EA
.:1B37 EA A9 0F 8D 0E
.:1B3C 90 A9 DC 8D 0C
.:1B41 90 A9 00 A2 00
.:1B46 A0 00 20 DB FF
.:1B4B EA 20 DE FF C9
.:1B50 08 D0 F9 A9 00
.:1B55 8D 0E 90 A9 00
.:1B5A 8D 0C 90 60 EA
.:1B5F EA A9 00 A2 00
.:1B64 A0 00 20 DB FF
.:1B69 EA 20 DE FF C9
.:1B6E 02 D0 F9 60 00
```

BLOCKCHECK

```
5 REMV71583
10 PRINT"▣   ▣  BLOCK  CHECK"
20 PRINT"▣ * (C) 1983 PSIDAC *"
40 PRINT"▣▣▣ ▣     IN PROGRESS▣▣▣▣"
100 DATA8192,16384,24576,40960,1024,0
110 READAD:CO=CO+1:IX=0:IY=0:IFCO=6THENGOTO260
120 FORI=1TO25:POKEAD+R,I:IFPEEK(AD+R)<>ITHENIX=1
122 POKEAD+R,0:IFCO<5THENR=R+180
125 R=R+120:NEXT:IZ=IZ+IX:R=0
130 IFIX=1THEN150
140 GOTO210
150 X=PEEK(AD):FORI=1TO100:IFPEEK(AD)<>XTHENIY=1
155 NEXT:IZ=IZ+IY
160 AC=0:FORZ=0TO6400STEP256:IFPEEK(AD+Z)=(AD+Z)/256THEN AC=AC+1
165 NEXTZ:IFAC=26THENIY=1:IZ=IZ+IY
170 IFIY=1THEN110
180 IFCO=4THENPRINT"ROM IN BLOCK 5":GOTO110
190 IFCO=5THENPRINT"ROM IN 3K EXPANSION   AREA":GOTO110
200 PRINT"ROM IN BLOCK"CO:GOTO110
210 IFCO=4THENPRINT"RAM IN BLOCK 5":GOTO110
220 IFCO=5THENPRINT"RAM IN 3K EXPANSION   AREA":GOTO110
230 PRINT"▣RAM IN BLOCK▣"CO:GOTO110
250 IFIZ=10THENPRINT"NO RAM/ROM FOUND"
260 IFIZ=10THENPRINT"NO RAM/ROM FOUND"
270 PRINT"▣▣▣         DONE▣":END
```

APPENDIX A

THE BINARY NUMBER SYSTEM

We are familiar with the base ten number system.

It utilizes ten digits 0-9.

We will study the binary number system or base two.

It uses two digits 0 & 1.

A binary digit is often called a bit.

We can represent any value in base *ten by agreeing that each place has a value or weight *ten times that of the one to its right thus:

Value of Place	1000	100	10	1	
Number	1,	2	4	7	7 x 1
					+ 4 x 10
					+ 2 x 100
					+ 1 x 1000
Value represented =					1,247

This elementary point will help us understand any number system.

Any value in base *two can be expressed by agreeing that each place has a value *2 times greater than the one right. (Notice that all bases begin with one) Thus, the binary number 1001 has a value of 9 as shown:

Place Value	8	4	2	1	
Number	1	0	0	1	1 x 1 = 1
					+ 0 x 2 = 0
					+ 0 x 4 = 0
					+ 1 x 8 = 8
Value represented =					9

$$1\,0\,0\,1_{2} = 9_{10}$$

Study these two examples until it becomes clear how these number systems are organized.

Remember these rules hold true for all number systems.

We can now find the equivalent value of any binary number. Since processors usually work with 8 or 16 bit "words" (Binary #'s) we will restrict our work to these sizes.

Here is the largest 16 bit binary word.

Place Value -	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Number -	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The above 16 bit binary word = 65,535

Since each place has a one the total value is found by adding all the place values listed.

It is often helpful to be able to count in a number system to gain familiarity with it.

Study the count table below. Try to recognize the pattern in each number system.

TABLE A1

DECIMAL		BINARY				HEX		
10	1	8	4	2	1	256	16	1
	0	0	0	0	0	$		0
	1	0	0	0	1	$		1
	2	0	0	1	0	$		2
	3	0	0	1	1	$		3
	4	0	1	0	0	$		4
	5	0	1	0	1	$		5
	6	0	1	1	0	$		6
	7	0	1	1	1	$		7
	8	1	0	0	0	$		8
	9	1	0	0	1	$		9
1	0	1	0	1	0	$		A
1	1	1	0	1	1	$		B
1	2	1	1	0	0	$		C
1	3	1	1	0	1	$		D
1	4	1	1	1	0	$		E
1	5	1	1	1	1	$		F
1	6	-	-	-	-	$	1	0

BINARY WORKSHEET

PROBLEM SET 1

Complete this sheet and check answers in the back of this section.

Find Base 10 value of these:

Example: 1 0 1 1 0 1 0 1
= 1 + 4 + 16 + 32 + 128
= 181

1) 1 1 1 1
=

2) 1 0 1 0
=

3) 0 0 1 0
=

4) 0 1 1 1
=

5) 0 1 0 1
=

6) 1 1 0 0 0 0 0 1
=

7) 0 0 1 1 1 1 0 0
=

8) 1 0 1 1 0 1 1 0
=

9) 0 1 0 1 0 1 0 1
=

10) 0 0 1 0 0 1 1 1
=

11) 0 0 0 0 0 0 1 1
=

Notice that eight bit binary words which we will call <u>BYTES</u> (say bites) correspond to the eight bit data and instruction words that we know as machine language.

Often we will wish to enter a decimal number into the processor. It must first be converted to a binary number. Here is one easy method.

Assume 237_{10} is to be converted.

Remember the place values for binary

256	128	64	32	16	8	4	2	1
0	1	1	0	0	0	0	0	0

Binary Equivalent

1) Determine which place value will go into 237 only one time. (In this case it is 128)

2) Put a one under the 128 as shown above.

3) Subtract 128 from 237 and repeat process.

   237-128 = 109

   64 will go into 109 only once so put a one under 64

   109-64 = 45

   Repeat with 32

4) If a next lower value will not go into a remainder put a zero in this spot and continue. If it goes more than once something went wrong!

BINARY WORKSHEET

PROBLEM SET 2

Complete and check answers

Find Binary equivalent of these

(Show as 1 byte words) (8 bits). Check against answers in back.

Example: 65

66 - 64 = 2

0 1 0 0 0 0 1 0

1) 135

2) 8

3) 15

4) 29

5) 7

6) 50

7) 255

8) 72

9) 13

10) 157

THE HEXADECIMAL NUMBER SYSTEM

We remember that binary was chosen as a universal digital language because it is compatible with electronic switching techniques. This is fine for machines that have static memories but for us poor biological creatures, we find that long strings of binary numbers are difficult to recognize and painful to write and talk about.

Base 16 is compatible with binary and easy to remember. The main feature of "hex" is that 4 bits of binary can be translated directly to a hex digit.

Base 16 has 16 counting digits: 0 - 9 and A - F. We must use letters because we cannot allow a single digit to hold two places. Notice table A1. "$A" corresponds to decimal 10 and "$10" to decimal 16. Notice that it also has a place value system of 1, 16, 256, etc.

To convert binary to hex we first separate the 8 bit word into two four bit sections:

$$\underbrace{1\ 0\ 0\ 1}_{9}/\underbrace{1\ 1\ 0\ 0}_{C}$$

Next look up hex values of the four bit numbers in table A1.

9 C = 1 0 0 1 1 1 0 0

The opposite is done for hex to binary.

HEXADECIMAL WORKSHEET

PROBLEM SET 3

Complete these and check answers.

Convert to opposite base (2 or 16)

Example:

Given:	5	B	1 0 1 0 /	0 0 0 1
	0 1 0 1	1 0 1 1	A	1

1) F C

2) A 0

3) 2 1

4) D B

5) E 8

6) 7 6

7) 1 1 0 1 1 1 1 1

8) 1 0 0 0 1 0 0 0

9) 0 0 0 0 1 1 1 0

10) 0 1 0 0 1 0 0 1

11) 1 1 1 0 1 0 1 1

12) 1 0 1 0 1 1 0 1

You should now be able to convert between: base 10, base 2, and base 16. The program HEXDECON is listed at the end of this appendix. It allows you to enter a number in hex or decimal and will convert it to the opposite.

BITS & BYTES

Now that we have learned how to work with binary and hex numbers, let's see how they relate to machine language.

A bit is either a one which electrically means a "HI" voltage. (+ 5 V for TTL ckt), or zero or "LO" voltage (0 V or ground). These voltages are present on the 8 wire data bus forming an 8 bit word or byte. The processor looks at data in 1 byte groups. Look at the example on the hex worksheet. Notice that 8 bits corresponds to 2 hex digits. Thus, 5 B is a one byte word which the processor might use as data or an instruction. This byte is one machine language word.

2 bytes	1 1 1 1 1 0 0 0	0 1 1 1 1 0 0 1	Binary for processor
	F 8	7 9	Hex for us

Sometimes 2 bytes are used for big numbers or memory. If this is done, all 16 bits form one large number. However, we must separate these bytes for many computers which can only handle 8 bit words. When this is done we call the first half (bits 15 - 8) the high order byte. Bits 7 - 0 make up the low order byte.

ANSWERS TO PROBLEM SETS

SET 1.	1)	15	6)	193	11)	3
	2)	10	7)	60		
	3)	2	8)	182		
	4)	7	9)	85		
	5)	5	10)	39		
SET 2.	1)	10000111	6)	00110010		
	2)	00001000	7)	11111111		
	3)	00001111	8)	01001000		
	4)	00011101	9)	00001101		
	5)	00000111	10)	10011101		
SET 3.	1)	11111100	7)	DF		
	2)	10100000	8)	88		
	3)	00100001	9)	0E		
	4)	11011011	10)	49		
	5)	11101000	11)	EB		
	6)	01110110	12)	AD		

```
10000 REM (C) PSIDAC 82
10005 PRINT"□□    HEX-DEC-CON":FORTD=1TO1000:NEXT
10010 PRINT"□ *******□MENU *******□"
10020 PRINT"1=HEX TO DEC"
10030 PRINT"2=DEC TO HEX□"
10040 INPUT"CHOICE";CH$
10050 IFCH$="1"THEN10100
10060 IFCH$="2"THEN10400
10070 PRINT"ILLEGAL CHOICE":FORTD=1TO1000:NEXT
10080 GOTO10010
10100 PRINT"□HEX TO DEC CON.□"
10110 INPUT"HEX";X$
10120 IFX$="EXIT"THEN10010
10130 D$(1)=LEFT$(X$,1)
10140 D$(2)=MID$(X$,2,1)
10150 D$(3)=MID$(X$,3,1)
10160 D$(4)=RIGHT$(X$,1)
10170 FORI=1TO4
10175 D(I)=VAL(D$(I))
10180 IFD$(I)="A"THEND(I)=10
10190 IFD$(I)="B"THEND(I)=11
10200 IFD$(I)="C"THEND(I)=12
10210 IFD$(I)="D"THEND(I)=13
10220 IFD$(I)="E"THEND(I)=14
10230 IFD$(I)="F"THEND(I)=15
10240 NEXT
10250 D(1)=D(1)*4096
10260 D(2)=D(2)*256
10270 D(3)=D(3)*16
10280 PRINT"DEC="D(1)+D(2)+D(3)+D(4)
10290 GOTO10110
10400 PRINT"□DEC TO HEX CON.□"
10410 INPUT"DEC";X$:X=VAL(X$)
10420 IFX$="EXIT"THEN10010
10430 D(1)=INT(X/4096)
10435 IFD(1)>15THENPRINT"OVERFLOW ERROR":GOTO10410
10440 X=X-(D(1)*4096)
10450 D(2)=INT(X/256)
10460 X=X-(D(2)*256)
10470 D(3)=INT(X/16)
10480 D(4)=X-(D(3)*16)
10490 FORI=1TO4
10500 D$(I)=STR$(D(I))
10510 IFD$(I)=" 10"THEND$(I)=" A"
10520 IFD$(I)=" 11"THEND$(I)=" B"
10530 IFD$(I)=" 12"THEND$(I)=" C"
10535 IFD$(I)=" 13"THEND$(I)=" D"
10538 IFD$(I)=" 14"THEND$(I)=" E"
10540 IFD$(I)=" 15"THEND$(I)=" F"
10550 NEXT
10560 PRINT"HEX="D$(1)D$(2)D$(3)D$(4)
10570 GOTO10410
```

APPENDIX B

PARTS AND SOFTWARE

For your convenience the following Parts, Kits, and Software can be ordered directly from PSIDAC.

ORDER #	DESCRIPTION	PRICE
PT-1	Pirate's tape; tape version of Romulator, Savmach, Hdrswap, Hexdecon, and Blockcheck.	$ 9.95
TK-1	Tapeworm Kit; Parts and PC board	12.95
TK-PC	Tapeworm PC board only	3.95
TK-CM	Tapeworm Parts only	9.95
RMK-1	Romulator Kit; Parts and PC boards (includes 8K RAM switch card)	14.95
RS-1	8K RAM switch card	3.50
SC-1	Super Clone plug kit	5.95
CONN	Six pin edge connector for VIC	2.50
PA-1	PIRATES ARSENAL Pirate's tape, Tapeworm Kit, Romulator Kit, Super Clone Kit	29.95
SHIRT	"THE SOFTWARE PIRATES" T -shirt S-M-L state size	9.95

Shipping add 10% to order total. ($3.00 maximum)

Order from:

PSIDAC KITS
[redacted] Portland, OR 97217

Check or Money order only. Personal checks allow 1 week extra for bank clearance. Make payable to PSIDAC.

ABOUT THE AUTHORS

David Thom and Vic Numbers have been involved in a working association for over six years. They met at an educational institution where both serve as electronics technology instructors.

Vic Numbers has an extensive background in electronics. He has 15 years of experience in custom designed automatic test systems used for fault analysis at a Naval weapons testing facility.

David Thom has been involved in work with microprocessor applications for electro-mechanical systems as well as video games. Mr. Thom also served as Engineering Manager for Windmills International Diversified over a two year period.

Since their association, Numbers and Thom have been involved in several industrial projects including automated advertising display and electronic reader boards. Thom and Numbers also headed an engineering team in designing a computerized intertie controller for a 150KW wind electrical generator.

As educators, Numbers and Thom became intrigued by the latent potential of the Vic-20. After cracking some of the secrets of the Vic, they realized that many people would appreciate the added usefulness that this information would give their computers. With this in mind, they set out to write this book for all Vic-20 users.

THE SOFTWARE PIRATE'S HANDBOOK FOR THE VIC-20 IS A COMPREHENSIVE GUIDE FOR MAKING BACKUP COPIES OF YOUR PROGRAMS. SPECIAL ATTENTION IS GIVEN TO LAWS AND ETHICS GOVERNING SOFTWARE DUPLICATION. THE BOOK EXPLAINS CONCEPTS RELATING TO SOFTWARE PROTECTION THAT CAN BE USED WITH THE VIC-20. SOFTWARE AND HARDWARE AIDS FOR BACKUP PURPOSES ARE FULLY DOCUMENTED INCLUDING SCHEMATICS, PC LAYOUTS, AND PROGRAM LISTINGS. STEP-BY-STEP PROCEDURES ARE GIVEN TO SIMPLIFY THE MORE DIFFICULT PROCESSES.
THE HANDBOOK COVERS TECHNIQUES FOR 8K and 16K CARTRIDGES AS WELL AS TECHNIQUES FOR ALL TAPED SOFTWARE.

A PSIDAC PUBLICATION [redacted] **ORTLAND, OREGON 97217**

Sugg. Retail
$9.95 U.S.